САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

РЕКОМЕНДАЦИИ ПО ПРЕПОДАВАНИЮ ИНФОРМАТИКИ В УНИВЕРСИТЕТАХ

Computing Curricula 2001: Computer Science

С.-Петербург
2002

УДК 37.022
ББК 74.58
Р36

Переводчики: М.Е. Зверинцева, Т.В. Зверинцева, Н.Ю.Курочка, А.А. Симановский, Д.А. Шапоренков
Редакторы перевода: В.Л. Павлов, А.А. Терехов

Рекомендовано к изданию кафедрой системного программирования С.-Петербургского государственного университета

Рекомендации по преподаванию информатики в университетах: Пер. с англ. — СПб., 2002. — 372 с.
Р36
ISBN 5-288-03105-3

Книга представляет собой заключительный отчет специальной объединенной комиссии ACM и IEEE Computer Science, содержащий рекомендации по преподаванию информатики и типовым учебным планам этой дисциплины.

Книга будет полезна преподавателям и студентам в области информатики.

Без объявл. ББК 74.58

ISBN 5-288-03105-3

Содержание

ОТ РЕДАКТОРОВ ПЕРЕВОДА

Что такое информатика? Как преподавать информатику? Какими чертами должен обладать специалист в области информатики? Эти вопросы всегда волновали российскую научную общественность. Одним из первых эти вопросы поднял еще в 70-80-х годах XX века академик А.П. Ершов в своих знаменитых статьях "Откуда берутся люди, способные создавать надежное программное обеспечение", "Программирование – вторая грамотность" и "О предмете информатики".

Последовавшее бурное обсуждение вопросов преподавания информатики привело к созданию и внедрению государственных образовательных стандартов, отражавших существовавшее на тот момент понимание предмета информатики и соответствующих ему знаний. Наиболее распространенным стандартом обучения информатике был учебный план по специальности 220400, описанный в статье С.С. Лаврова, А.О. Слисенко и Г.С. Цейтина "Проект учебного плана специальности: "информатика и системное программирование" ("Микропроцессорные средства и системы", №4, 1985). Эта специальность просуществовала без значительных изменений до 2000 года, когда она была заменена похожей по содержанию специальностью 351500.

Интересно отметить, что российские исследования в области обучения информатике развивались в значительной мере самостоятельно и в некотором отрыве от мировых разработок. В то же время зарубежные усилия по стандартизации обучения информатике заслуживают самого пристального внимания. Самым значительным проектом в этой области, безусловно, является создание и обновление документа Computing Curricula ("Рекомендации по преподаванию информатики в университетах").

Первая версия Computing Curricula была разработана специальным комитетом по образованию профессионального общества ACM (Association for Computing Machinery) и вышла в свет в 1968 году. В 70-х годах аналогичный документ выпустило другое профессиональное общество – IEEE Computer Society. Наконец, в конце 80-х годов эти организации объединили свои усилия и в 1991 году выпустили обновленную версию рекомендаций – Computing Curricula'91.

Именно с этой версии документа началось наше знакомство с "Рекомендациями по преподаванию информатики". Мы попытались соотнести рекомендации этого документа с учебным планом обучения информатике в Санкт-Петербургском государственном университете. Результаты оказались весьма обескураживающими – к 1992 году наша программа обучения не накрывала и 40% курсов, описанных в Compputing Curricula. С того момента ситуация определенно улучшилась, но все-таки мы и сегодня не можем сказать, что наша программа целиком соответствует этому стандарту.

В 2001 году появился черновой вариант новой версии Computing Curricula. Окончательный вариант Computing Curricula 2001 был опубликован в декабре 2001 и сразу же после этого наш коллектив начал работать над русской версией текста.

Сложности, с которыми нам пришлось столкнуться, работая над документом, легко проиллюстрировать даже на примере его названия. Краткое и емкое английское словосочетание "Computing Curricula" разрослось до "Рекомендаций по преподаванию информатики в университетах". Надо сказать, что все прочие обсуждавшиеся варианты перевода названия были еще более громоздкими или недостаточно точно отражали суть рекомендаций этого отчета. В дальнейшем тексте отчета мы иногда употребляем и другие варианты названия – "Рекомендации по составлению учебных планов по информатике" или "Рекомендации к учебным планам по информатике".

Еще хуже обстояло дело с переводом терминологии – для многих из использованных в отчете терминов еще не появилось устоявшегося русского перевода. В других случаях выяснялось, что мы недостаточно хорошо разбираемся в предметной области для того, чтобы дать точный перевод содержания рекомендуемых курсов (например, в области компьютерной графики). Иногда нам удавалось решить проблемы перевода с помощью различных англо-русских словарей, которыми мы пользовались при переводе, но, к сожалению, достаточно часто мы сталкивались с новыми или специализированными терминами, не переведенными ни в одном из словарей. В таких случаях мы обращались к специалистам в данной предметной области. Наконец, мы старались приводить в скобках английский оригинал в сложных или потенциально неоднозначных случаях.

В заключение, мы хотели бы поблагодарить компании и организации, которые помогли нам в издании этой книги:

- Ассоциация Предприятий Компьютерных и Информационных Технологий (АП КИТ, Россия), генеральный спонсор проекта
- ЗАО "ЛАНИТ-ТЕРКОМ" (Санкт-Петербург, Россия)
- Украинско-американское СП Information Systems Development (Днепропетровск, Украина)
- Фирма eLine Software (Питтсбург, США)
- Кафедра системного программирования Санкт-Петербургского государственного университета

Без поддержки этих организаций данный проект просто не состоялся бы. С помощью спонсоров проекта удалось обеспечить не только подготовку русской версии Computing Curricula 2001, но и финансирование рассылки книги в ведущие технические университеты России и Украины.

Мы просим читателей с пониманием отнестись к тому факту, что тираж издания ограничен, и обращаем внимание, что электронная версия этого документа свободно доступна в Интернете по адресу http://se.math.spbu.ru/cc2001. Английский оригинал отчета можно найти по адресу: http://www.computer.org/education/cc2001.

В.Л. Павлов (ISD), vlpavlov@ieee.org
А.А.Терехов (ЛАНИТ-ТЕРКОМ, СПбГУ), terekhov@computer.org

Состав объединенной специальной комиссии по созданию Рекомендаций по учебным планам преподавания 2001

Вице-президент, комитет по образованию IEEE-CS
Carl Chang
Председатель, комитет по образованию ACM
Peter J. Denning

Представители IEEE-CS
James H. Cross II (сопредседатель)
Gerald Engel (сопредседатель и редактор)
Robert Sloan (секретарь)
Doris Carver
Richard Eckhouse
Willis King
Francis Lau
Susan Mengel
Pradip Srimani

Представители ACM
Eric Roberts (сопредседатель и редактор)
Russell Shackelford (сопредседатель)
Richard Austing
C. Fay Cover
Gordon Davies
Andrew McGettrick
G. Michael Schneider
Ursula Wolz

Одобрено советом ACM, ноябрь 2001 г.
Одобрено советом управляющих IEEE-CS, декабрь 2001 г.

ПРЕДИСЛОВИЕ

Этот документ представляет собой заключительный отчет по проекту "Рекомендации по преподаванию информатики 2001" (Computing Curricula 2001, в дальнейшем – CC2001), который является результатом совместной работы Компьютерного общества Института инженеров по электротехнике и электронике (IEEE-CS) и Ассоциации по вычислительной технике (ACM). В рамках этого проекта было разработано руководство по составлению учебных планов по информатике для университетов и колледжей. Данный отчет продолжает давно существующую серию рекомендаций – первый такой документ появился еще в 1965 году (см. главу 2 настоящего отчета).

В этом томе отчета приведен набор рекомендаций для университетских программ в области информатики (computer science). Как указано ниже в главе 1, в конечном итоге отчет CC2001 будет состоять из нескольких томов, содержащих отдельные рекомендации для различных дисциплин информатики, включая проектирование компьютеров (computer engineering), программную инженерию (software engineering), а так же информационные системы (information systems). Все эти отчеты разрабатываются соответствующими комиссиями, и будут публиковаться по мере их завершения.

Ключевыми моментами данного отчета являются:

- *Совокупность знаний по информатике.* Мы определили набор знаний, который должны покрывать университетские программы по информатике. Основываясь на структуризации, использованной в предыдущих отчетах, мы упорядочили эту совокупность знаний в стройную иерархию, подразделяя всю дисциплину на отдельные области знаний, которые, в свою очередь, подразделяются на разделы и отдельные темы. Обзор совокупности знаний по информатике приводится в главе 5.

- *Обязательный набор курсов по информатике.* Из 132 тем в совокупности знаний мы выбрали 64, представляющие собой обязательный материал ("ядро" знаний по информатике), рассчитанный приблизительно на 280 часов обучения. Как сказано в формулировке наших основных принципов в главе 4, мы определяем обязательный курс как набор разделов, признаваемый большинством преподавателей как необходимый для студентов, изучающих информатику. Обоснование подобного определения обязательного набора курсов приведено в главе 5.

- *Задачи обучения.* Для каждого из блоков в наборе знаний мы определили набор задач обучения, предназначенных для объективизации оценки достижений студентов. Эти цели обучения представлены как часть детального описания совокупности знаний в Приложении А. В дополнение к индивидуальным целям обучения, глава 11 данного отчета выделяет более общие цели, стоящие перед выпускниками факультета информатики.

- *Модели изложения материала.* В отчете определены шесть различных подходов к изложению вводного курса по информатике. Эти подходы, доказавшие свою успешность на практике, описаны в главе 7. Глава 8 предлагает четыре тематических подхода к изложению материала основных курсов. Обсуждение моделей изложения материала продолжается в главе 9, в которой предлагается несколько моделей для учебных планов преподавания в целом.

- *Описания курсов.* Приложение Б содержит детальное описание 47 курсов, входящих в различные учебные модели. Кроме того, мы перечислили более 80 дополнительных углубленных курсов, которые могут преподаваться в университетских программах.

Процесс разработки отчета был достаточно трудоемким. Более 150 человек были непосредственно вовлечены в рабочие группы, созданные в рамках проекта. Кроме того, весь отчет (как его предварительные версии, так и окончательный вариант) был тщательно просмотрен преподавателями и профессионалами. Наконец, мы провели целый ряд специальных семинаров на различных конференциях и совещаниях, включая симпозиум специальной группы по образованию в области информатики (SIGCSE), конференцию "Передний край в образовании" (FIE), всемирный конгресс по вопросам в области информатики и образования (WCCE), а также различные совещания меньших масштабов в Европе, Азии и различных частях Соединенных Штатов. Эти совещания дали нам столь необходимую обратную связь. Мы постарались учесть полученные замечания при создании заключительного варианта отчета.

ГЛАВА 1. ВВЕДЕНИЕ

Осенью 1998 года Компьютерное сообщество Института Инженеров по Электротехнике и Электронике (IEEE-CS) и Ассоциация по Вычислительной Технике (ACM) основали специальную комиссию по учебным планам преподавания информатики с целью пересмотра существовавших на тот момент руководств по составлению учебных планов для университетских программ по информатике. Официально задача была сформулирована в уставе комиссии следующим образом:

> *Пересмотреть рекомендации ACM и IEEE-CS по учебным планам преподавания информатики 1991 года, и разработать исправленную и дополненную версию для 2001 года, которая будет учитывать самые последние достижения компьютерных технологий за последнее десятилетие и которая сможет выдержать проверку временем в течение последующего десятилетия.*

Как определено в нашем уставе, целью работы комиссии по созданию отчета CC2001 является пересмотр предыдущей версии отчета, "*Рекомендаций по составлению учебных планов преподавания информатики 1991"* путем включения в документ результатов, полученных в последнее десятилетие. Эта задача оказалась намного более сложной, чем мы изначально думали. Информатика существенно изменилась, и эти изменения сильно повлияли на структуру учебных планов и педагогику. Более того, границы того, что мы называем *информатикой*, настолько расширились, что становится трудно определять ее как единую дисциплину. В предыдущих отчетах предпринимались попытки объединить такие дисциплины как информатика, проектирование компьютеров и программная инженерия в рамках одного отчета о компьютерном образовании. Хотя подобный подход был разумным 10 лет назад, сейчас не возникает сомнений, что в XXI веке информатика состоит из целого ряда самостоятельных дисциплин, каждая из которых имеет свою педагогическую специфику.

1.1. Общая структура документов CC2001

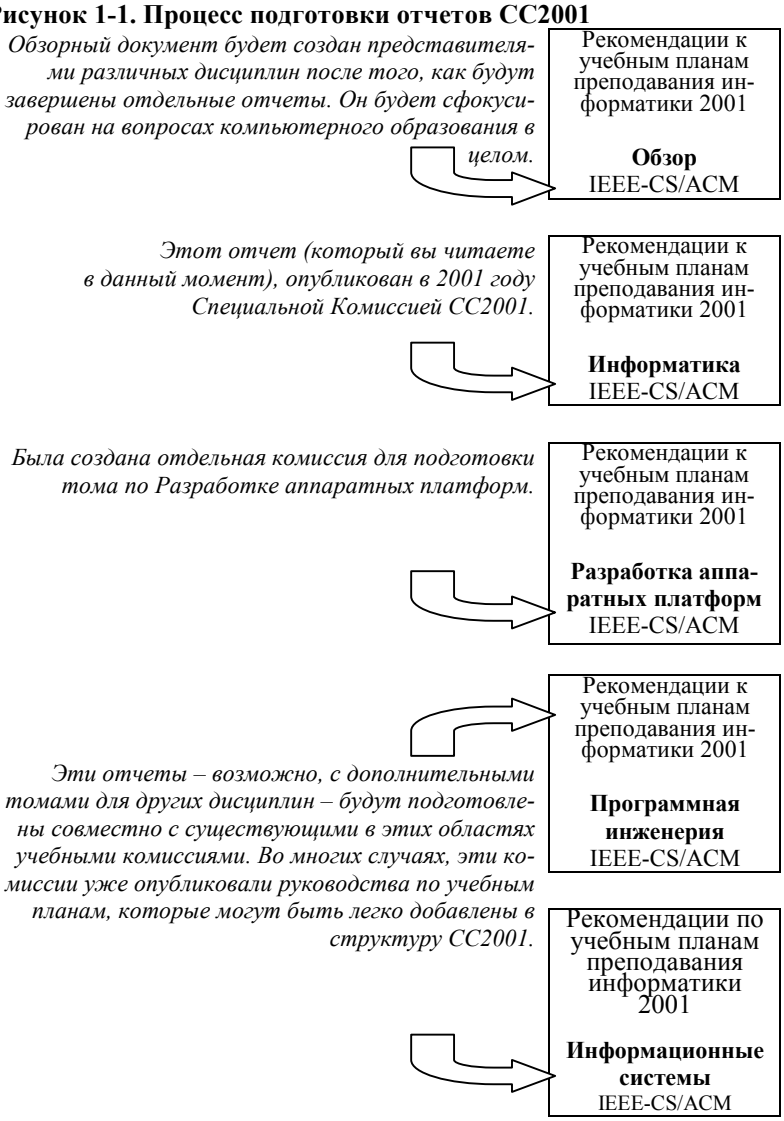
В свете расширения дисциплины информатики – и поскольку обратная связь, которую мы получили после публикации чернового варианта отчета, подтолкнула нас к движению в этом направлении – мы решили разделить отчет CC2001 на несколько частей. Данный том посвящен только информатике (computer science). Однако для того, чтобы охватить и другие дисциплины, являющиеся частью обобщенной области информатики и информационных технологий, IEEE-CS и ACM создали ряд дополнительных комиссий. Эти комиссии предпринимают аналогичные шаги в других областях, включая проектирование компьютеров, программную инженерию и информационные системы.

Когда отдельные отчеты будут завершены, представители всех дисциплин соберутся вместе, чтобы создать обзорный том, объединяющий всю серию отчетов. Этот обзорный том будет содержать определения различных дисциплин информатики вместе с анализом общих приемов преподавания. Структура серии проиллюстрирована на рисунке 1-1.

1.2. Обзор процесса разработки CC2001

Разработка данного документа была преимущественно осуществлена специальной комиссией CC2001, члены которой перечислены в начале этого документа. Учитывая масштабы проекта CC2001 и объем рассматриваемого материала, мы сочли необходимым привлечь множества дополнительных специалистов, представляющих различные учреждения и области знаний. Для обеспечения участия широкого круга участников, необходимого для успеха таких проектов, специальная комиссия образовала 20 рабочих групп, разделенных на две категории: *рабочие группы по структуризации знаний* (Knowledge Focus Groups, KFGs) и *рабочие группы по педагогике* (Pedagogy Focus Groups, PFGs).

Рисунок 1-1. Процесс подготовки отчетов CC2001



1.2.1. Рабочие группы по структуризации знаний

В самом начале работы над данным документом, специальная комиссия CC2001 определила 14 областей, на которые делится совокупность знаний по информатике (см. рис. 1-2). Для каждой из этих областей была назначена рабочая группа по структуризации знаний, состоявшая из экспертов с опытом преподавания в данной сфере. Рабочим группам было поручено подготовить отчеты по своей области, из которых специальная комиссия могла бы составить полную совокупность знаний по информатике. Детальное изложение этого процесса приведено в главе 5 и приложении A.

Рисунок 1-2. 14 рабочих групп по структуризации знаний

Дискретные структуры (DS)
Основы программирования (PF)
Алгоритмы и теория сложности (AL)
Архитектура и организация ЭВМ (AR)
Операционные системы (OS)
Распределенные вычисления (NC)
Языки программирования (PL)
Графика и визуализация (GV)
Интеллектуальные системы (IS)
Управление информацией (IM)
Социальные и профессиональные вопросы программирования (SP)
Программная инженерия (SE)
Методы вычислений (CN)
Человеко-машинное взаимодействие (HC)

1.2.2. Рабочие группы по педагогике

Хотя рабочие группы по структуризации знаний необходимы для спецификации совокупности знаний в каждой поддисциплине, сами по себе они не могут считаться достаточными для разработки учебных планов. Так как каждая рабочая группа рассматривает информатику сквозь призму своей конкретной области, структура KFG не поддерживает создания широкого видения учебного плана, основанного на изложении "всепроникающих" тем (т.е. тем, изучаемых сразу в нескольких областях информатики). Для того чтобы разработать учебный план более целостно и охватить все многообразие вопросов, преодолевающих границы индивидуальных поддисциплин, спе-

циальная комиссия CC2001 образовала шесть рабочих групп по педагогике для изучения проблем обучения, относящихся к информатике в целом. Рабочие группы по педагогике, вместе с их конкретными задачами, перечислены на рисунке 1-3.

Рабочие группы по педагогике были созданы позднее, чем рабочие группы, занимающиеся структуризацией областей знаний. Более того, оказалось, что работу групп по педагогике намного труднее организовывать средствами электронной телекоммуникации. Благодаря щедрому гранту Национального научного фонда США (National Science Foundation), специальная комиссия CC2001 получила возможность собрать очное совещание рабочих групп по педагогике в июне 2000 года. Это совещание внесло существенный вклад в процесс разработки заключительных отчетов.

Рисунок 1-3. Шесть рабочих групп по педагогике и их обязанности

PFG1. Вводные темы и курсы

a. Определить цели первого года обучения.
b. Изучить сильные и слабые стороны традиционного подхода "Вначале программирование".
c. Предоставить краткий список альтернативных подходов, а также анализ их соответствия задачам обучения.
d. Определить и/или разработать одну или более последовательностей вводных курсов, направленных на решение проблемы различий в подготовке поступающих студентов и не допускающих расслоения студентов по знаниям, полученным в школе, а также представляющих информатику как базовую дисциплину, формирующую часть основных знаний у широкого круга поступающих студентов.
e. Представить краткий список альтернатив для первого года изучения информатики, который отвечает целям пункта (a) и который может служить моделью для колледжей и издателей.

PFG2. Дополнительные темы и курсы

a. Определить ряд образовательных целей, необходимых для студента помимо собственно информатики, например, знание математики, инженерии, науки, навыков создания технической документации и публичных выступлений, знание экономики и управления проектами и т.п.
b. Определить минимальный список вспомогательных тем, считающихся важными в любом базовом курсе обучения информатики независимо от типа учебного института.
c. Представить на рассмотрение предложения по ассортименту дополнительных (поверх минимума) и вспомогательных тем, который может варьироваться в зависимости от типа института, аудитории, на которую ориентирован институт, и количества курсов, которое институт имеет возможность включить в программу.
d. Разработать детали для одного или более наборов некомпьютерных курсов, которые подходят для достижения целей пункта (a).
e. Разработать одну или более моделей для достижения нескольких либо всех целей пункта (a) путем их интегрирования в компьютерные курсы.

PFG3. Фундамент информатики

a. Используя спецификацию обязательных знаний в области информатики, разработать небольшое количество моделей изложения учебного материала, которые удовлетворяли бы основным требованиям. Каждая модель должна состоять из краткого списка курсов (4-5 курсов помимо начального года обучения), который должен быть прочитан каждому выпускнику факультета информатики и который мог бы подходить для практически любого типа базовой университетской программы.
b. Разработать как минимум одну модель изложения материала, которая являлась бы альтернативой традиционному подходу организации программ вокруг артефактов (например, курсы по компиляторам, операционным системам и т.п.). Подобные модели будут состоять из пересекающихся курсов, посвященных "всеобщим" фундаментальным концепциям, принципам и навыкам, актуальным во многих областях программирования.
c. Разработать как минимум одну модель изложения материала, при которой Интернет является главной объединяющей темой.

PFG4. Профессиональная практика

a. Описать те аспекты профессиональной практики, которые наши выпускники получают или должны получать в результате обучения в университете.
b. Описать, что мы знаем и не знаем о путях повышении эффективности выработки практических навыков у студентов.
c. Описать, как выработка практических навыков может быть интегрирована в курсы в учебных планах.
d. Описать влияние работы в промышленных условиях и стажировок на выработку у студентов практических навыков.
e. Описать другие аспекты профессионализма (включая этические, социальные, юридические и моральные принципы) и их взаимосвязь с базовым учебным планом по информатике.

PFG5. Углубленное обучение и исследовательская работа

a. Основываясь на определении "ядра" информатики как дисциплины, разработать спецификацию по образованию в области информатики за пределами "ядра", необходимую и достаточную для университетских курсов по информатике.
b. Разработать подробное описание требований к выпускникам, получившим степень за четыре года обучения (степень магистра).
c. Включить описание курсов, – как в традиционных, так и в нетрадиционных областях – которые могут быть важны для современного учебного плана преподавания информатики.
d. Описать основные направления исследовательской работы, включая оценку различных существующих ее моделей.

PFG6. Информатика в учебных планах

a. Четко выделить "основу основ" информатики, необходимую для всех студентов и для различных семейств академических дисциплин.
b. Спланировать и разработать надлежащий подход к разработке учебных планов, предназначенных для студентов, не специализирующихся в области информатики, и студентов тех институтов, которые отличаются от традиционных четырехгодичных университетов (например, двухгодичные программы обучения в США). Такие учебные планы должны также быть интересны студентам, обучающимся по специальностям, связанным с информатикой или активно использующим информатику.
c. Признать, что это чрезвычайно важная задача, для которой мы не можем самостоятельно разработать адекватное решение.
d. Признать, что задачей данной рабочей группы является не разрешение проблемы, но планирование, разработка, и инициализация процесса, который может привести и в конечном итоге приведет к решению.

1.2.3. Специальная комиссия по вопросам двухгодичных колледжей

После выпуска в начале 2001 года предварительного варианта CC2001, образовательный комитет по вопросам двухгодичных колледжей ACM сформировал специальную комиссию ACM/IEEE-CS, чьей целью было создание дополнительного отчета по учебным планам для двухгодичных колледжей. Эта специальная комиссия провела детальное изучение черновых вариантов отчета CC2001, уделяя особое внимание вступительным темам по информатике и связанным с ними целям обучения, объему математических знаний и факультативных курсам начального уровня. Работа специальной комиссии по вопросам двухгодичных колледжей существенно повлияла на отчет CC2001, в частности, эта работа заложила основу для итоговых вариантов вводных курсов. Тем самым, нам удалось обеспечить большую совместимость между двух- и четырехлетними учебными планами преподавания, упрощая процесс перевода студентов с двухлетних курсов на обычные университетские программы.

1.3. Структура отчета по информатике CC2001

Данный том отчета CC2001 рассматривает только информатику. Основная часть отчета состоит из 13 глав. Глава 2 начинается с обзора и анализа более ранних отчетов, прежде всего, концентрируясь на *Рекомендациях к учебным планам преподавания информатики 1991*. Глава 3 описывает основные изменения, которые произошли в информатике с момента публикации отчета CC1991, и влияние этих изменений на преподавание и разработку учебных планов. В главе 4 мы выделяем ряд принципов, которыми мы руководствовались при разработке CC2001, пытаясь использовать достижения предыдущих попыток и избегать их недостатков. Главы 5 и 6 представляют обзоры совокупности знаний по информатике и рекомендаций к учебным планам, детально рассмотреных в приложениях. Главы 7 и 8 описывают курсы и подходы, которые мы рекомендуем на вводном и основном уровнях учебного плана соответственно. Так как эти курсы сами по себе не могут составлять полный учебный план, глава 9 посвящена дополнительным курсам и темам, которые должны быть включены в учебные программы университетов. Важным аспектом полного учебного плана является профессиональная практика, которая подробно обсуждается в главе 10. В главе 11 мы выделяем ряд качеств, характеризующих успешных выпускников курса информатики. Глава 12 рассматривает проблему преподавания информатики и выработки навыков, связанных с информатикой, у студентов других дисциплин. И, наконец, глава 13 представляет стратегические и тактические предложения по борьбе с институциональными проблемами, влияющими на реализацию рекомендаций этого отчета.

Основная часть материала данного отчета представлена в двух приложениях. Приложение А детально рассматривает совокупность знаний, представляемую университетским курсом по информатике. Приложение Б состоит из подробных описаний рекомендованных курсов, приведенных в типовом учебном плане. Мы надеемся, что предоставление в данном документе совокупности знаний и описаний курсов сделает процесс создания учебных планов более эффективным, чем при использовании лишь одного из этих источников.

ГЛАВА 2. УРОКИ ПРЕДЫДУЩИХ ОТЧЕТОВ

При разработке этого отчета специальной комиссии CC2001 не пришлось начинать с нуля. Мы извлекли огромную пользу из предыдущих исследований по составлению учебных планов и признательны авторам этих исследований за приложенные ими усилия. Как часть нашей работы по подготовке *Рекомендаций к учебным планам преподавания информатики 2001*, мы внимательно рассмотрели самые последние исследования по созданию университетских программ – в частности *Рекомендации к учебным планам преподавания информатики 1991* – для того, чтобы понять, как эти исследования повлияли на образование в сфере информатики. Определяя, какие аспекты предыдущих отчетов были успешными, а какие нет, мы надеялись по-новому структурировать отчет CC2001 с целью увеличения его воздействия. Эта глава предлагает обзор предыдущих отчетов и уроков, которые мы извлекли из них.

2.1. Исторические сведения

Первые попытки создания типовых учебных планов для факультетов по информатике и проектированию компьютеров были предприняты в 1960-х годах, сразу после появления первых факультетов в этих областях. В 1968 году, вслед за серией ранних исследований [ACM65, COSINE67, SAC67], ACM опубликовала *Учебный план'68* [ACM68], в котором были приведены детальные рекомендации для учебных программ по информатике, а также набор описаний курсов и огромная библиография для каждой тематической области.

На протяжении следующего десятилетия компьютерная наука быстро развивалась и в какой-то момент рекомендации *Учебного плана'68* стали целиком устаревшими. В 1970-х годах и ACM, и Компьютерное Сообщество IEEE (IEEE-CS) создали комитеты по разработке исправленных и уточненных учебных планов по информатике. В 1977 году комитет по вопросам образования IEEE-CS опубликовал отчет для программ по компьютерной науке и проектированию [EC77]. Отчет IEEE-CS был содержательным с той точки зрения, что он продемонстрировал более широкий взгляд на дисциплину, внося больше инженерных вопросов в учебную программу и сокращая разрыв между программами, ориентированными на программное обеспечение и программами, ориентированными на аппаратное обеспечение. Реагируя на быстрое развитие данной области, IEEE-CS обновило свой отчет в 1983 году [EAB 83]. Отчет ACM *Учебный план'68* был заменен на расширенную версию *Учебный план'78*, которая имела значительное влияние на образование в сфере информатики. В частности, в *Учебном плане'78* был предложен ряд стандартных курсов, описывающих основные знания информатики.

В конце 1980-х годов IEEE-CS и ACM объединили свои силы для проведения более амбициозного проекта по пересмотру учебных планов. Результат этих работ был опубликован под названием *Рекомендации по составлению учебных планов по информатике 1991* [Tucker91], далее по тексту CC1991. Отчет CC1991 был более полным, чем его предшественники, но его подход был другим. В отличие от *Учебного плана'78* и отчета IEEE-CS за 1983 год, каждый из которых был сконцентрирован на определении содержания стандартного набора курсов, CC1991 разделил совокупность знаний по информатике, на отдельные *разделы знаний (knowledge units)*. Каждый раздел знаний в CC1991 соответствует некоторой теме, которая должна быть изложена в университетских курсах обучения, хотя учебным заведениям предоставлена значительная свобода в группировке разделов знаний в свои планы преподавания согласно своим специфическим потребностям. Приложение к отчету CC1991 включало в себя 11 примеров реализации, которые показывали, как разделы знаний могут быть скомбинированы в курсы и программы самых разнообразных назначений.

2.2. Оценка предыдущих подходов к разработке учебных планов

Решение создать новый отчет по учебным планам возникло в первую очередь благодаря существенным изменениям, которые произошли в компьютерной науке за последние десять лет. В то же время, среди преподавателей информатики получило распространение мнение, что CC1991 был менее влиятельным, чем его предшественники. Хотя отчет CC1991 определенно более подробен, чем остальные рекомендации, многие институты обнаруживали, что реализовать его намного труднее, чем *Программу обучения'78* или учебную модель IEEE-CS по информатике и вычислительной технике.

Для определения сильных и слабых сторон CC1991, специальная комиссия провела неформальный опрос среди преподавателей информатики. Мы разработали и разослали на кафедры всех факультетов информатики в Соединенных Штатах и Канаде краткую анкету. Кроме того, мы сделали эту анкету доступной в Интернет, хотя подавляющее большинство ответов все равно было из Северной Америки. Копия анкеты представлена на рисунке 2-1.

Рисунок 2-1. Анкета для оценки влияния отчета CC1991

1. Использовали ли вы в прошлом CC1991 для каких-либо целей?
2. Если вы преподаватель колледжа или университета, известно ли вам, использовался ли или хотя бы рассматривался CC1991 на вашем факультете?

3. Если вы положительно ответили на первый или второй вопрос, то как был использован отчет CC1991 и какие его разделы были полезными?
4. Считаете ли вы необходимым создание CC2001? Почему?
5. В CC1991 было определено 10 основных областей знаний. Считаете ли вы, что в этот список должны быть добавлены какие-либо новые области? Считаете ли вы, что какие-либо из существующих областей должны быть убраны или модернизированы?
6. Считаете ли вы, что CC2001 должен предоставлять рекомендации по минимальному объему обязательного материала? Если да, то что должен включать в себя этот минимум ("ядро")?
7. Есть ли у вас какие-либо предложения относительно формата? CC1991 был разработан в терминах разделов знаний вместе с возможными модельными учебными планами, построенными на базе этих разделов знаний.
8. Есть ли у вас какие-либо еще комментарии или предложения относительно обновления CC1991?

Мы получили 124 ответа на нашу анкету через Интернет и около 30 ответов посредством обычной почты. Более 98 процентов опрошенных поддержали концепцию обновления отчета CC1991. Результаты опроса также показали, что:

- *Разделы знаний зачастую менее полезны, чем курсы или проекты курсов обучения.* Многие респонденты указали, что им нравится концепция разделов знаний в качестве ресурса, но в то же время большинство респондентов предлагали сделать больший акцент на разработке курсов, соответствующих разделам знаний. Наш опрос показал, что многие институты продолжают работать в соответствии с учебными моделями, описанными в *Учебном плане'78*, в основном из-за того, что в этот документ были включены конкретные проекты курсов.

- *Существует потребность в более точном определении минимального набора обязательных курсов (минимального "ядра" информатики).* CC1991 требовал, что все университетские программы по информатике включали все разделы знаний из девяти областей информатики, определенных как обязательные. Если область "Введение в языки программирования" включена в программу, то это означает добавление к учебной программе 283 часов аудиторного времени. По мере развития нашей дисциплины возникает желание добавлять все больше материала в обязательные требования, что приведет к существенному увеличению количества часов в учебных программах. Наш опрос обнаружил значительную поддержку идеи сужения списка тем, обязательных для изучения. Тематика же и структура дополнительных курсов могут заметно варьироваться в зависимости от рода института, учебной программы, а так же нужд и интересов каждого из студентов.

- *Отчеты по учебным планам должны уделять больше внимание критериям аккредитации программ по информатике.* Аккредитация была важным вопросом для многих респондентов в Соединенных Штатах. Однако необходимо заметить, что структура аккредитации заметно изменилась с появлением новых критериев, предложенных Комитетом по аккредитации программ в области инженерии и технологий (ABET) и Комитетом по аккредитации программ в области компьютерных наук (CSAB) [ABET2000, CSAB2000]. В соответствии с новыми руководствами, в программах допускается больше гибкости, чем ранее, но при этом необходимо предоставлять обоснование для своих учебных планов и подтверждать достижение намеченных целей. Данный отчет составлен не только для того, чтобы помочь институтам спроектировать свои учебные программы по информатике, но также и для помощи в подготовке обоснований этих программ, необходимых для удовлетворения новых критериев аккредитации. Мы также надеемся, что этот отчет окажется полезным органам аккредитации и в других странах мира.

ГЛАВА 3. ИЗМЕНЕНИЯ В ИНФОРМАТИКЕ КАК ДИСЦИПЛИНЕ

К началу нового тысячелетия информатика стала чрезвычайно актуальной и популярной областью. С момента своего появления около пятидесяти лет назад, информатика стала определяющей технологией нашего времени. Компьютеры превратились в неотъемлемую часть современной культуры, и являются движущей силой экономического роста во всем мире. Более того, эта область продолжает развиваться с поразительной скоростью. Постоянно появляются новые технологии, а существующие технологии становятся устаревшими практически сразу после возникновения.

Быстрая эволюция дисциплины оказала сильное воздействие на образование в области информатики, влияя как на содержание преподаваемых дисциплин, так и на педагогические методы. Например, во время публикации отчета CC1991, сетевые технологии не воспринимались как самостоятельная тема – им было отведено только шесть часов из списка общеобязательных. Это и неудивительно, так как тогда использование сетей еще не было массовым явлением, а WWW был не более чем набором идей в умах ее создателей. Сегодня сетевые технологии и WWW стали основой для большой части нашей экономики. Они стали необходимым фундаментом компьютерной науки, и сегодня уже невозможно представить себе программу обучения информатики, в которой этой теме не уделялось бы больше внимания. В то же время, существование WWW изменило природу самого образовательного процесса. Современные сетевые технологии улучшают способность общения каждого человека и предоставляют людям во всем мире небывалый доступ к информации. В большинстве учебных программ на сегодняшний день – не только в информатике, но также и в других областях – сетевые технологии стали важным педагогическим инструментом.

Устав специальной комиссии CC2001 требует от нас "пересмотра рекомендаций ACM и IEEE-CS по учебным планам преподавания информатики 1991 года, и разработки исправленной и дополненной версии для 2001 года, которая будет учитывать самые последние достижения компьютерных технологий за последнее десятилетие". Для выполнения этой задачи мы посчитали необходимым потратить часть наших усилий на исследование того, какие аспекты информатики изменились за последнее десятилетие. Как нам кажется, эти изменения делятся на две категории, техническую и культурную, каждая из которых оказывает существенное влияние на образование в сфере информатики. Основные изменения в каждой из этих категорий описаны ниже в отдельных разделах.

3.1. Технические изменения

Многие изменения, влияющие на информатику, связаны с прогрессом в технологии. Большинство этих достижений представляют собой часть постоянного эволюционного процесса, который длится уже многие годы. Закон Мура (прогноз, сделанный в 1965 году создателем Intel Гордоном Муром, и гласящий, что плотность транзисторов на кристалле микропроцессора будет удваиваться каждые восемнадцать месяцев) до сих пор является истинным. В результате, мы наблюдаем экспоненциальный рост вычислительных возможностей, благодаря которым стало возможным решение задач, которые казались неразрешимыми всего лишь несколько лет назад. Другие, еще более впечатляющие изменения в дисциплине, такие как быстрый рост сетей после появления World Wide Web, показывают, что изменения могут носить и революционный характер. Как эволюционные, так и революционные изменения влияют на объем минимального набора знаний, обязательного для изучения в рамках программ по информатике.

Технические достижения за последнее десятилетие увеличили важность многих учебных тем, в частности:

- WWW и его приложения
- Сетевые технологии, в частности, базирующиеся на TCP/IP
- Графика и мультимедиа
- Встроенные системы
- Реляционные базы данных
- Способность к взаимодействию (interoperability)
- Объектно-ориентированное программирование
- Использование программных интерфейсов приложения (API)
- Человеко-машинное взаимодействие
- Надежность программного обеспечения
- Безопасность и криптография
- Конкретные предметные области (application domains)

С увеличением значимости этих тем, появляется естественное желание поместить их в списки обязательных курсов. К сожалению, ограничения большинства учебных программ не позволяют свободно добавлять новые темы без удаления старых. Зачастую невозможно охватить новые области без сокращения времени, предназначенного для более традиционных тем, важность которых постепенно ослабевает с течением времени. Поэтому специаль-

ная комиссия CC2001 попыталась сократить требуемый объем изложения большинства изучаемых тем, с тем, чтобы освободить пространство для новых. Этот вопрос обсуждается подробнее в главе 4.

3.2. Культурные изменения

На компьютерное образование также влияют изменения в культурном и социальном контексте. Например, все из перечисленных ниже изменений повлияли на природу образовательного процесса:

- *Изменения в педагогике в результате появления новых технологий.* Технические изменения, которые привели к расширению информатики, напрямую влияют и на культуру обучения. Например, компьютерные сети сделали дистанционное образование намного более доступным, приведя к существенному развитию этой области. Кроме того, компьютерные сети намного облегчили совместное использование учебных ресурсов географически распределенными институтами. Технология также влияет и на педагогику. Демонстрационное программное обеспечение, компьютерные проекторы и персональные компьютеры привели к значительным изменениям в преподавании информатики. Структура курсов по информатике должна учитывать эти изменения в технологии.

- *Неожиданная скорость распространения компьютеров во всем мире.* Компьютеры получили чрезвычайное распространение в течение последнего десятилетия. Например, в 1990 году очень немногие семьи – даже в США – имели домашнее подключение к Интернету. В 1999 году Министерство торговли США обнаружило, что уже более трети всех американцев имели доступ в Интернет [NTIA99]. Такой же рост наблюдается и в других странах. Бурное распространение компьютерных технологий приводит к множеству изменений, влияющих на обучение, включая и общее увеличение уровня осведомленности студентов в области информатики и ее прикладных задач. Однако в то же время увеличивается разрыв между уровнем знаний тех, кто имеет доступ к современным компьютерным технологиям и тех, кто такого доступа не имеет.

- *Растущее экономическое влияние компьютерных технологий.* Повышенный общественный интерес к индустрии высоких технологий (доказательством которого служит, хотя бы недавняя "золотая лихорадка" вокруг так называемой Интернет-экономики) существенно повлиял на образование и выделяемые для него ресурсы. Огромный спрос на профессионалов в области информатики и надежда сколотить себе состояние в области информатики привлекают все большее количество студентов в эту сферу, в том числе и тех, кому информатика по существу не интересна. В то же время, растущий спрос на специалистов со стороны коммерческих компаний усложнил для большинства институтов привлечение и удержание преподавателей, тем самым значительно ограничив возможности институтов удовлетворять потребность рынка в молодых специалистах.

- *Увеличивающееся признание информатики как академической дисциплины.* В свои ранние годы информатика была вынуждена отстаивать свою легитимность во многих учебных заведениях. В конце концов, это была новая дисциплина без глубоких исторических корней, характерных для большинства академических наук. В некотором роде, эта проблема существовала даже во время создания отчета CC1991, который был тесно связан с отчетом "Информатика как дисциплина" [Denning89]. Во многом в результате внедрения компьютерных технологий в основные культурные и экономические аспекты нашей жизни, борьба за легитимность была выиграна. Во многих учебных заведениях информатика стала одной из самых крупных и активных дисциплин. Больше нет никакой необходимости в отстаивании обучения информатике в высших учебных заведениях. Сегодня основной проблемой является нахождение путей удовлетворения спроса на такое обучение.

- *Расширение дисциплины.* Наша дисциплина не только выросла и стала легитимной, но и значительно расширила свои границы. В ранние годы информатика (computing) во многом сводилась к компьютерной науке (computer science). С годами, все больше и больше областей становились частью информатики. Специальная комиссия CC2001 считает, что понимание взаимосвязей между различными областями информатики, осознание степени расширения дисциплины и влияния этих факторов на образование должны стать существенным компонентом нашей работы.

ГЛАВА 4. ПРИНЦИПЫ

Основываясь на анализе предыдущих отчетов по учебным планам и изменений в дисциплине информатики, изложенных в предыдущих главах, специальная комиссия CC2001 сформулировала следующие руководящие принципы своей работы:

1. *Информатика (computing) – это широкая область исследований, которая не может быть сведена к рамкам компьютерной науки (computer science).* Одного отчета, охватывающего только "чистую" информатику, недостаточно для того, чтобы описать весь спектр вопросов, встающих перед колледжами и университетами при создании учебных планов по компьютерным дисциплинам. Для полноценного охвата всех областей и направлений информатики необходимо создать серию специальных отчетов.

2. *Информатика основывается на целом ряде дисциплин.* Университетское обучение информатике требует от студентов использования концепций из множества разнообразных областей. Все студенты, изучающие информатику, должны учиться объединять теорию и практику, понимать важность обобщения и абстракции, а также ценить хорошие инженерные решения.

3. *Быстрая эволюция компьютерной науки требует постоянного пересмотра учебных планов.* Учитывая темп изменений в нашей дисциплине, обновление учебной программы раз в десять лет уже не является приемлемым. Профессиональные организации в области информатики должны организовать постоянный процесс пересмотра типовых учебных планов, который позволит оперативно обновлять устаревшие компоненты.

4. *При разработке типовых учебных планов по информатике необходимо учитывать изменения в технологиях, новые разработки в сфере педагогики, а также все возрастающую важность обучения на протяжении всей жизни (life-long learning).* В такой быстро развивающейся области как информатика, учебные заведения должны оперативно перенимать передовые стратегии, реагируя на происходящие изменения. Учебные заведения должны не отставать от прогресса как в области технологий, так и в области педагогики, даже несмотря на существующие ограничения в ресурсах. Кроме того, обучение информатике в институте должно готовить студентов к дальнейшему (само)обучению на протяжении всей жизни, что позволит им двигаться в ногу со временем и быть способными разрешать сложные проблемы будущего.

5. *Отчет CC2001 не должен ограничиваться описанием разделов знаний – необходимо также предложить набор рекомендаций по разработке отдельных курсов.* Хотя структура разделов знаний, которая использовалась в CC1991, может служить полезной основой, большинство институтов нуждается в более детальном руководстве. Для таких институтов CC2001 будет полезным только в том случае, если в нем будет определен небольшой (от двух до четырех) набор альтернативных моделей, который упорядочивает разделы знаний в рационально построенные и легко реализуемые курсы. Идентификация четко определенных моделей упростит для институтов обмен педагогическими стратегиями и средствами. Это также даст полезную концепцию издателям, выпускающим учебники и другие материалы для этих курсов.

6. *CC2001 должен определить базисные навыки и знания, которыми должны обладать все студенты, специализирующиеся по информатике.* Несмотря на значительное расширение компьютерной науки, существуют концепции и навыки, которые являются общими для информатики в целом. CC2001 должен попытаться определить общие темы дисциплины, и описать их в рамках базовой программы.

7. *Набор обязательных для изучения знаний должен быть уменьшен настолько, насколько это возможно.* По мере расширения дисциплины информатики, количество тем, обязательных для изучения, заметно увеличилось. За последнее десятилетие информатика разрослась до такой степени, что теперь уже невозможно добавлять новые темы без удаления старых. Мы считаем, что в таких условиях лучшей стратегией является *сокращение* количества тем в наборе обязательных знаний. Поэтому мы определили минимальный набор обязательных курсов, включающий в себя только тот материал, который практически все преподаватели информатики признают необходимым для студентов, желающих получить диплом в области информатики. В то же время необходимо понимать, что обязательные курсы сами по себе не могут составить полноценной учебной программы по информатике. Поэтому все учебные планы должны включать дополнительные факультативные разделы (разделы по выбору), хотя данный отчет и не определяет, какие именно. Факультативные разделы, скорее всего, будут отличаться в зависимости от конкретного учебного учреждения, специализации и личных предпочтений каждого студента.

8. *CC2001 должен быть полезным для всего мирового сообщества.* Несмотря на то, что требования к учебным планам преподавания различаются от страны к стране, CC2001 должен быть полезным для преподавателей информатики во всем мире. Очевидно, что на отчет CC2001 существенно повлияет образовательная практика в Соединенных Штатах, но мы должны приложить все усилия к тому, чтобы наши рекомендации учитывали национальные и культурные различия и были применимы во всем мире.

9. *В разработку CC2001 должно быть вовлечено максимальное количество заинтересованных лиц.* Для достижения успеха, к созданию рекомендаций CC2001 должны привлекаться представители различных заинтересованных сторон, в том числе, промышленности, правительственных органов и всего диапазона высших учебных заведений, обучающих информатике.

10. *CC2001 должен включать в себя профессиональную практику в качестве неотъемлемой компоненты университетского учебного плана.* Практическая работа студента должна включать в себя широкий спектр видов деятельности, таких как менеджмент, развитие этики и моральных ценностей, письменное и устное общение, работа в команде, постоянное изучение последних достижений в быстро меняющейся дисциплине. Мы подтверждаем и развиваем позицию, заявленную в отчете CC1991: "овладение навыками информатики включает в себя не только глубокое знание и понимание дисциплины, но и умение применить ее концепции к проблемам реального мира".

11. *Отчет CC2001 должен включать в себя обсуждение различных стратегий и тактики реализации учебных программ наравне с рекомендациями высокого уровня.* Хотя данный отчет должен выражать наше видение проблем обучения информатике, успех любого учебного плана в большой степени зависит от деталей реализации. CC2001 должен помогать институтам в практическом создании учебного плана. Для этого в отчет необходимо включить разделы по стратегии и тактике реализации, включающие необходимый уровень технических деталей.

Оглядываясь сегодня на результаты проделанной работы, становится очевидно, что некоторые принципы были реализованы специальной комиссией CC2001 в большей степени, чем другие (чего, впрочем, и следовало ожидать от проекта подобного масштаба). Например, мы не достигли того уровня "интернационализации" документа, на который мы рассчитывали. Структура высшего образования очень существенно варьируется от страны к стране и, наверное, невозможно выработать единый набор рекомендаций, который работал бы во всем мире. Хотя мы и включили в главу 9 примеры реализации учебных планов, созданных для использования в других странах, структура преподавания информатики в Соединенных Штатах глубоко повлияла на отчет. Кроме того, мы не смогли получить достаточной обратной связи и поддержки от промышленности. Однако мы рассматриваем разработку учебных планов как постоянный процесс и надеемся, что коммерческие компании будут лучше участвовать в нем на уровне сотрудничества с отдельными учебными заведениями.

В то же время, мы считаем, что выполнили свои обязательства по минимизации объема обязательного набора знаний на управляемом, но достаточном уровне, обеспечивающем наличие у выпускников необходимого фундамента. Более того, мы уверены, что материал Приложений А и Б предоставляет достаточно детальной информации о материале и структуре учебных курсов, чтобы быть полезным для разработчиков учебных программ во всем мире.

ГЛАВА 5. ОБЗОР СОВОКУПНОСТИ ЗНАНИЙ ПО ИНФОРМАТИКЕ

При разработке учебных планов по информатике для университетов, одним из первых шагов является определение и организация материала, который соответствовал бы необходимому уровню преподавания. Как отмечено в главе 1, специальная комиссия CC2001 старалась достичь этой цели путем организации нескольких рабочих групп, поставив перед ними задачу определения минимального набора обязательных знаний в каждой из следующих областей:

- Дискретные структуры (DS)
- Основы программирования (PF)
- Алгоритмы и теория сложности (AL)
- Архитектура и организация ЭВМ (AR)
- Операционные системы (OS)
- Распределенные вычисления (NC)
- Языки программирования (PL)
- Взаимодействие человека и машины (HC)
- Графика и визуализация (GV)
- Интеллектуальные системы (IS)
- Управление информацией (IM)
- Социальные и профессиональные вопросы программирования (SP)
- Программная инженерия (SE)
- Методы вычислений (CN)

Каждая из рабочих групп по структуризации знаний представила свой отчет на рассмотрение специальной комиссии CC2001, которая затем определила, являются ли рекомендации, выработанные той или иной группой, значимыми в контексте информатики в целом.

5.1. Структура совокупности знаний

Совокупность знаний по информатике организована в виде трехуровневой иерархической структуры. На верхнем уровне иерархии находится **область**, представляющая собой отдельную часть дисциплины информатики. Каждая область обозначается двухбуквенной аббревиатурой, например, OS для *операционных систем* или PL для *языков программирования*. Области делятся на меньшие структуры, называемые **разделами**, которые представляют собой отдельные тематические модули внутри области. Каждый раздел обозначается численным суффиксом, добавляемым к имени области, например, OS3 обозначает раздел *параллелизма*. Каждый раздел, в свою очередь, состоит из набора **тем**, представляющих собой нижний уровень этой иерархии.

Основные и факультативные разделы

Как обсуждалось в главе 4, информатика как дисциплина настолько расширилась, что студенты уже не могут освоить все темы, которые когда-либо считались фундаментальными. Поэтому комиссия решила определить минимальный набор **обязательных** курсов, включающий в себя только тот материал, который практически все преподаватели информатики признают необходимым для студентов, желающих получить диплом в области информатики. Материал, выходящий за рамки данного набора, рассматривается как **факультативный** (разделы по выбору). Настаивая на максимально распространенном определении основного набора знаний, комиссия надеется сохранить этот набор как можно более малым, давая тем самым образовательным учреждениям свободу в выборе факультативных компонент учебной программы с учетом их индивидуальных потребностей.

Обсуждая рекомендации CC2001 в процессе их разработки, мы пришли к выводу, что полезно подчеркнуть следующие соображения:

- *Обязательный материал состоит из тех разделов, знание которых требуется от всех студентов, изучающих информатику.* Некоторые темы, которые являются важными в образовании многих студентов, не включены в обязательный материал. Отсутствие темы в наборе обязательного материала не означает негативного суждения о ее ценности, важности либо релевантности. Это просто означает, что не было всеобщего согласия о том, что данная тема необходима *каждому* студенту в *каждой* программе обучения компьютерной специальности.
- *Обязательные курсы сами по себе не являются полной учебной программой.* Поскольку набор обязательных курсов по определению является минимальным, он не может считаться полной учебной программой.
- *Обязательные курсы должны дополняться факультативными материалами.* Любая учебная программа должна включать факультативные разделы совокупности знаний. Данный отчет не определяет содержание

этих курсов, так как эта дополнительная работа может и должна различаться в зависимости от задач университета, основных областей исследований данного института и личных предпочтений студентов.

- *Обязательные разделы не обязаны ограничиваться набором вводных курсов, читаемых на ранних стадиях учебной программы.* Хотя многие из обязательных разделов действительно являются вводными по своему содержанию, имеется также несколько обязательных разделов, требующих для своего освоения солидный объем предварительных знаний. Например, комиссия полагает, что на каком-то этапе обучения все студенты должны самостоятельно разработать сложное приложение. Поэтому обязательный материал включает в себя сведения по управлению проектами – эти знания должны получить все студенты. Однако обычно такой проект выполняется ближе к концу обучения. Сходным образом, вводные курсы могут включать факультативные разделы, примыкающие к материалу курсов. Таким образом, термин *"обязательный курс"* ничего не говорит о времени чтения курса.

Оценка времени, необходимого для изучения раздела

Для того чтобы дать читателям представление о времени, необходимом для изучения отдельного раздела, документ CC2001 определяет стандартные метрики. Выбор такой метрики оказался сложной задачей, так как не существует общепринятой меры этой величины. Для согласования с ранними версиями документа, комиссия решила измерять время в часах, что соответствует аудиторным часам, необходимым для представления материала в традиционном формате, ориентированном на лекции. Во избежание непонимания, однако, важно подчеркнуть следующие наблюдения, касающиеся нашего выбора единиц измерения:

- *Комиссия не ставит своей задачей рекомендовать лекционный формат.* Хотя мы использовали метрику, основанную на классическом, лекционном стиле, комиссия уверена, что существуют другие методы, которые являются, по меньшей мере, столь же эффективными. Для многих из этих методов понятие *учебного часа* может оказаться не вполне адекватным. Но даже в этом случае временные характеристики могут послужить хотя бы в качестве меры сравнения, в том смысле, что 5-часовой раздел будет предположительно занимать в пять раз больше времени, чем 1-часовой, независимо от стиля преподавания.
- *Указываемые часы не включают в себя время, проводимое вне аудитории.* Время, отводимое на раздел, не включает в себя время подготовки преподавателя и время, затрачиваемое студентами вне аудитории. В качестве рекомендации заметим, что объем внеаудиторных занятий должен примерно в три раза превосходить объем аудиторных. Так, раздел, требующий 3 часа должен обычно изучаться 12 часов (3 часа в аудитории и 9 часов самостоятельно).
- *Указываемые часы, отводимые на раздел, подразумевают минимальный объем сведений.* Временные показатели, отведенные нами для каждого раздела, должны пониматься как минимальное количество времени, требуемое студенту для освоения раздела в рамках, требуемых программой. Всегда допустимо и полезно отводить на раздел больше времени, чем обязательный минимум.

5.1.3. Организация разделов знаний в курсы

Структура и формат курсов изменяется в зависимости от института и страны. Даже в Соединенных Штатах некоторые колледжи и университеты используют семестровую систему, в то время как другие делят год на четверти. Однако при любой системе количество недель в семестре, количество лекций в неделю и даже количество минут в лекции может немного отличаться в различных университетах.

В целях данного отчета, мы предположили, что **курс** читается три раза в неделю в течение 15-недельного семестра и что отдельные аудиторные занятия занимают от 50 минут до часа. Такой график типичен для трехкредитного семестрового курса в Соединенных Штатах. Учитывая, что некоторая часть времени семестра тратится на экзамены и другую деятельность, мы получили, что в течение семестра доступно около 40 лекционных часов. В дополнение к этому, ожидается, что студенты должны тратить по три часа внеаудиторного времени на каждый аудиторный час. Это означает, что суммарное время, которое студенты будут затрачивать на один курс, должно равняться примерно 160 часам. В других странах используются другие системы измерения для выражения ожидаемого объема работы. Например, в Великобритании курс, описанный выше, будет приравниваться к 15-16 пунктам в соответствии со схемой накопления и передачи зачетных баллов.

5.2. Краткое описание совокупности знаний по информатике

Краткий обзор совокупности знаний по информатике приведен на рисунке 5-1. Подробное описание приведено в Приложении А.

Рисунок 5-1. Рисунок А-1. Совокупность знаний по информатике (обязательные темы подчеркнуты)

DS. Дискретные структуры (43)
DS1. Функции, отношения и множества (6)
DS2. Основы логики (10)
DS3. Методы доказательства (12)
DS4. Основы вычислений (5)
DS5. Графы и деревья (4)
DS6. Дискретная вероятность (6)

PF. Основы программирования (38)
PF1. Основные конструкции программирования (9)
PF2. Алгоритмы и решение задач (6)
PF3. Фундаментальные структуры данных (14)
PF4. Рекурсия (5)
PF5. Событийно-управляемое программирование (4)

AL. Алгоритмы и теория сложности (31)
AL1. Основы анализа алгоритмов (4)
AL2. Алгоритмические стратегии (6)
AL3. Фундаментальные вычислительные алгоритмы (12)
AL4. Распределенные алгоритмы (3)
AL5. Основы теории вычислимости (6)
AL6. Классы сложности P и NP
AL7. Теория автоматов
AL8. Углубленный анализ алгоритмов
AL9. Криптографические алгоритмы
AL10. Геометрические алгоритмы
AL11. Параллельные алгоритмы

AR. Архитектура и организация ЭВМ (36)
AR1. Цифровая логика и цифровые системы (6)
AR2. Представление данных в памяти компьютера (3)
AR3. Организация машины на уровне ассемблера (9)
AR4. Устройство памяти компьютера (5)
AR5. Взаимодействие и коммуникации (3)
AR6. Функциональная организация (7)
AR7. Многопроцессорные и альтернативные архитектуры (3)
AR8. Улучшение производительности
AR9. Архитектура сетевых и распределенных систем

OS. Операционные системы (18)
OS1. Обзор операционных систем (2)
OS2. Основы операционных систем (2)
OS3. Параллелизм (6)
OS4. Планирование и диспетчеризация (3)
OS5. Управление памятью (5)
OS6. Управление устройствами
OS7. Безопасность и защита данных
OS8. Файловые системы
OS9. Встроенные системы и системы реального времени
OS10. Отказоустойчивость
OS11. Оценка производительности системы
OS12. Языки сценариев

NC. Распределенные вычисления (15)
NC1. Введение в распределенные вычисления (2)
NC2. Сети и телекоммуникации (7)
NC3. Сетевая безопасность (3)
NC4. Web как пример архитектуры "клиент-сервер" (3)
NC5. Разработка web-приложений
NC6. Управление сетями
NC7. Сжатие и распаковка данных
NC8. Технологии мультимедиа
NC9. Беспроводные и мобильные компьютеры

PL. Языки программирования (21)
PL1. Обзор языков программирования (2)
PL2. Виртуальные машины (1)
PL3. Введение в трансляцию (2)
PL4. Переменные и типы данных (3)
PL5. Механизмы абстракции (3)
PL6. Объектно-ориентированное программирование (10)
PL7. Функциональное программирование
PL8. Системы трансляции
PL9. Системы типов
PL10. Семантика языков программирования
PL11. Разработка языков программирования

HC. Взаимодействие человека и машины (8)
HC1. Основы взаимодействия человека и машины (6)
HC2. Построение простого графического интерфейса (2)
HC3. Оценка программного обеспечения, ориентированного на человека
HC4. Разработка программного обеспечения, ориентированного на человека
HC5. Проектирование графического интерфейса пользователя
HC6. Программирование графического интерфейса пользователя
HC7. Человеко-машинные аспекты мультимедиа-систем
HC8. Человеко-машинные аспекты сотрудничества и коммуникаций

GV. Компьютерная графика и визуализация (3)

GV1. Фундаментальные методы в графике (2)
GV2. Графические системы (1)
GV3. Графические коммуникации
GV4. Геометрическое моделирование
GV5. Основы рендеринга
GV6. Углубленное изучение рендеринга
GV7. Более сложные методы
GV8. Компьютерная анимация
GV9. Визуализация
GV10. Виртуальная реальность
GV11. Компьютерное зрение

IS. Интеллектуальные системы (10)

IS1. Основные вопросы, связанные с интеллектуальными системами (1)
IS2. Поиск решений (5)
IS3. Представление знаний и вывод (4)
IS4. Углубленное изучение поиска
IS5. Углубленное изучение представления знаний и вывода
IS6. Агенты
IS7. Обработка естественного языка
IS8. Обучение машины и нейронные сети
IS9. Системы искусственного интеллекта с планируемым поведением
IS10. Робототехника

IM. Управление информацией (10)

IM1. Информационные модели и системы (3)
IM2. Системы баз данных (3)
IM3. Моделирование данных (4)
IM4. Реляционные базы данных
IM5. Языки запросов к базам данных
IM6. Проектирование реляционных баз данных
IM7. Обработка транзакций
IM8. Распределенные базы данных
IM9. Проектирование физической структуры базы данных
IM10. Извлечение информации
IM11. Хранение и поиск информации
IM12. Гипертекст и гипермедиа
IM13. Мультимедийная информация и системы мультимедиа
IM14. Цифровые библиотеки

SP. Социальные и профессиональные вопросы (16)

SP1. История информатики (1)
SP2. Социальный контекст информатики (3)
SP3. Методы и средства анализа (2)
SP4. Профессиональная и этическая ответственность (3)
SP5. Недостатки компьютерных систем и риски, связанные с их применением (2)
SP6. Интеллектуальная собственность (3)
SP7. Конфиденциальность и гражданские свободы (2)
SP8. Компьютерные преступления
SP9. Экономические вопросы, связанные с применением компьютеров
SP10. Философские концепции

SE. Программная инженерия (31)

SE1. Проектирование ПО (8)
SE2. Использование программных интерфейсов приложений (5)
SE3. Программные средства и окружения (3)
SE4. Процессы разработки ПО (2)
SE5. Спецификации и требования к ПО (4)
SE6. Проверка соответствия ПО (3)
SE7. Эволюция ПО (3)
SE8. Управление программными проектами (3)
SE9. Компонентно-ориентированная разработка
SE10. Формальные методы
SE11. Надежность ПО
SE12. Разработка специализированных систем

CN. Вычислительная математика и численные методы (нет часов)

CN1. Численный анализ
CN2. Исследование операций
CN3. Моделирование
CN4. Высокопроизводительные вычисления

Замечание: Числа в скобках обозначают минимальное число часов, необходимое для изучения материала в лекционном формате. Всегда допустимо отводить больше времени.

ГЛАВА 6. ОБЗОР МОДЕЛЕЙ ИЗЛОЖЕНИЯ МАТЕРИАЛА

Совокупность знаний, представленная в главе 5, сама по себе не является учебным планом. Поэтому для того, чтобы быть полезным, отчет CC2001 должен также описать детальные реализации курсов и стратегии объединения отдельных курсов в завершенный учебный план. Эта глава представляет краткое описание общей философии, которая стоит за предлагаемыми моделями изложения материала. Описания самих курсов содержатся в Приложении Б.

6.1. Общая структура модельных учебных планов

Курсы, описанные в этом отчете, разделены на три категории в соответствии с уровнем, на котором они находятся в учебном плане. Курсы, определенные как *вводные*, являются типичными курсами начального уровня, которые предлагаются на первом или втором курсах обучения в колледже либо университете. Курсы, определенные как *основные*, являются курсами второго или третьего года и закладывают фундамент для дальнейшего изучения данной области. Курсы, обозначенные как *углубленные*, концентрируются на темах, требующих значительной предварительной подготовки на более ранних курсах.

Хотя данное разбиение достаточно ясно само по себе, важно не отождествлять уровень курса с понятиями *обязательный* и *факультативный (по выбору)*, которые относятся к разделам в совокупности знаний. Например, вводные и основные курсы определенно будут содержать преимущественно материалы обязательных разделов и все-таки полезно включать некоторые факультативные элементы уже в самые первые курсы. Аналогично, дополнительные курсы будут содержать некоторую часть материалов обязательных курсов. Таким образом, эти разделения независимы и их не следует путать.

6.2. Обзор стратегий составления учебного плана

Целью разграничения вводных, основных и углубленных является создание естественных рамок, в которых развивается составление учебного плана. Например, в данном отчете определяется шесть различных реализаций вводных курсов и четыре тематических подхода к компоновке основных курсов. Эти реализации и их взаимосвязь в структуре всего учебного плана показана на рисунке 6-1. Идея данной структуры заключается в том, чтобы предоставить разработчикам учебных планов большую гибкость, дав возможность начинать с любого вводного материала и затем использовать любой из подходов при разработке последовательности основных курсов.

Рисунок 6-1. Уровни курсов и стратегии обучения

6.3. Переход от стратегии к стратегии

Поскольку различные стратегии составления учебного плана используют различные подходы и охватывают различный материал, трудно добиться взаимозаменяемости последовательностей курсов различного типа. Чтобы дать институтам максимум гибкости, мы попытались уменьшить проблему взаимозаменяемости путем соблюдения следующих правил:

- Мы установили набор требований к вводным курсам в виде списка разделов и тем, которые должны охватываться в рамках любого из перечисленных выше подходов к формированию вводных курсов (подробно изложение этих требований приведено в главе 7). Учитывая существование таких требований, основные курсы всегда могут положиться на определенный уровень подготовки студентов, прошедших любую из последова-

тельностей вводных курсов. Данное описание знаний, общих для всех студентов, прослушавших вводные курсы, также должно облегчить институтам перевод студентов со специальности на специальность.

- Во всех учебных планах мы оставили некоторое незаполненное время в целях увеличения свободы преподавателей в выборе дополнительного материала, а также для обеспечения возможности включения материалов, необходимых для перехода от одного курса к другому.

- Мы сочли допустимым некоторое пересечение материалов, излагаемых на различных уровнях учебного плана. В случае если основной или углубленный курс зависит от материала, покрываемого некоторыми, но не всеми, вводными курсами, мы включали определенный объем этого материала в соответствующий основной или углубленный курс для обеспечения совместимости различных стратегий.

6.4. Охват обязательных курсов по информатике

Как показано на рисунке 6-1, полный учебный план состоит из вводной фазы (обеспечивающей базис для дальнейшего обучения), основной фазы (охватывающей большинство обязательных для изучения модулей) и дополнительных углубленных курсов, завершающих учебный план. Организации, решившие использовать модель CC2001, как правило, будут начинать с выбора вариантов реализации вводной и основной фаз. Затем необходимо провести адаптацию выбранных вариантов реализации под свои специфические потребности, учитывая конкретные характеристики организации, предпочтения преподавателей и пожелания студентов. При этом нужно убедиться, что итоговый учебный план включает в себя, по крайней мере, минимальный набор курсов, обязательных для изучения. Если какие-то из обязательных разделов не попали во вводную и основную фазы, институт должен обеспечить студентов возможностью изучения соответствующих материалов на углубленных курсах, и требовать обязательного знания этих материалов к моменту выпуска. Помимо изучения информатики, студенты должны также получить необходимый минимум знаний из других областей, как описано в главе 9.

Рисунки 6-2 и 6-3 показывают два примера того, как можно скомбинировать курсы из Приложения Б таким образом, чтобы они охватывали обязательные разделы информатики. Модель на рисунке 6-2 использует реализацию вводной фазы, ориентированную на императивное программирование, и традиционную модель тематического подхода для основных курсов; модель на рисунке 6-3 использует ориентированную на объектно-ориентированное программирование вводную стратегию и сокращенный подход для основного уровня. Конечно же, допустимы и иные комбинации. Для помощи в определении степени покрытия обязательного минимума тем или иным набором курсов, web-страница CC2001 содержит соответствующую программу проверки, реализованную в виде Java-апплета.

Таблицы, приведенные на рисунках 6-2 и 6-3, также иллюстрируют важность избыточного покрытия материала с точки зрения взаимозаменяемости отдельных моделей. Последняя колонка в каждой таблице показывает количество дополнительных часов, приходящихся на различные модули в рамках соответствующей комбинации курсов. Строка для PL3 (Введение в трансляцию) на рисунке 6-2, например, показывает, что два основных часа, предназначенных для этого блока, включены как в $CS112_I$, так и в $CS210_T$. При выборе данной пары стратегий, факультеты могут либо убрать этот материал из одного из курсов, освобождая время для дополнительных тем, либо включить его в оба курса, обеспечивая закрепление знания студентами данного материала.

Рисунок 6-2. Охват основных блоков. Введение, ориентированное на императивное программирование. Традиционный тематический подход.

	CS111. Введение в программирование	CS112. Представление данных	CS115. Дискретные структуры	CS210т. Анализ алгоритмов	CS220т. Архитектура ЭВМ	CS225т. Операционные системы	CS230т. Распределенные вычисления	CS260т. Искусственный интеллект	CS270т. Базы данных	CS280т. Соц. и проф. аспекты	CS290т. Разработка ПО	CS490. Дипломная работа	Сумма	Дополнительные часы
DS1. Функции, отношения и множества			6										6	
DS2. Основы логики			10										10	
DS3. Методы доказательства			9	3									12	
DS4. Основы вычислений			5										5	
DS5. Графы и деревья		2		4									6	+2
DS6. Дискретная вероятность			6										6	
PF1. Основные конструкции программирования	9												9	
PF2. Алгоритмы и решение задач	3			3									6	
PF3. Фундаментальные структуры данных	6	6		3									15	+1
PF4. Рекурсия		5											5	
PF5. Событийно-управляемое программирование							2				4		6	+2
AL1. Основы анализа алгоритмов		2		2									4	
AL2. Алгоритмические стратегии				6									6	
AL3. Фундаментальные вычислительные алгоритмы	2	4		6									12	
AL4. Распределенные алгоритмы						3							3	
AL5. Основы теории вычислимости	1			6									7	+1
AR1. Цифровая логика и цифровые системы			3		3								6	
AR2. Представление данных в памяти компьютера	1				3								4	+1
AR3. Организация машины на уровне ассемблера	2				9								11	+2
AR4. Устройство памяти компьютера					5								5	
AR5. Взаимодействие и коммуникации					3								3	
AR6. Функциональная организация					7								7	
AR7. Многопроцессорные и альтернативные архитектуры					3								3	
OS1. Обзор операционных систем						2							2	
OS2. Основы операционных систем						2							2	
OS3. Параллелизм						6							6	
OS4. Планирование и диспетчеризация						3							3	
OS5. Управление памятью						5							5	
NC1. Введение в распределенные вычисления							2						2	
NC2. Сети и телекоммуникации							7						7	
NC3. Сетевая безопасность							3						3	
NC4. Web как пример архитектуры "клиент-сервер"							3						3	
PL1. Обзор языков программирования	1	1											2	
PL2. Виртуальные машины		1											1	

PL3. Введение в трансляцию		2		2									4	+2
PL4. Переменные и типы данных	1	2											3	
PL5. Механизмы абстракции	2	1											3	
PL6. ОО-программирование	3	7					2						12	+2
HC1. Основы взаимодействия человека и машины									2		6	2	10	+4
HC2. Построение простого графического интерфейса											2		2	
GV1. Фундаментальные методы в графике	2										2		4	+2
GV2. Графические системы											1		1	
IS1. Основные вопросы интеллектуальных систем								1					1	
IS2. Поиск решений								5					5	
IS3. Представление знаний и вывод								4					4	
IM1. Информационные модели и системы									3				3	
IM2. Системы баз данных									3				3	
IM3. Моделирование данных									4				4	
SP1. История информатики	1									1			2	+1
SP2. Социальный контекст информатики										3			3	
SP3. Методы и средства анализа										2			2	
SP4. Профессиональная и этическая ответственность										3			3	
SP5. Риски и ответственность компьютерных систем										2			2	
SP6. Интеллектуальная собственность									3	3			6	+3
SP7. Конфиденциальность и гражданские свободы									2	2			4	+2
SE1. Проектирование ПО	2	2									2	4	10	+2
SE2. Использование программных интерфейсов приложений		2									3	3	8	+3
SE3. Программные средства и окружения	1	2									2	3	8	+5
SE4. Процессы разработки ПО												2	2	
SE5. Спецификации и требования к ПО	1										2	2	5	+1
SE6. Проверка соответствия ПО	1										1	3	5	+2
SE7. Эволюция ПО											2	2	4	+1
SE8. Управление программными проектами											2	3	5	+2
Сумма основных часов на каждый курс	**39**	**39**	**39**	**35**	**33**	**21**	**19**	**10**	**17**	**16**	**29**	**24**		

Рисунок 6-3. Охват основных блоков. Введение "С ориентацией на ОО программирование". Сокращенный подход.

	CS111$_O$. ОО-программирование	CS112$_O$. ОО проектирование	CS115. Дискретные структуры	CS210$_C$. Анализ алгоритмов	CS220$_C$. Архитектура ЭВМ	CS226$_C$. Операционные системы и сети	CS262$_C$. Управление инф-ей и данными	CS292$_C$. Разработка ПО и проф. практика	Сумма	Дополнительные часы
DS1. Функции, отношения и множества			6						6	
DS2. Основы логики			10						10	
DS3. Методы доказательства			9	3					12	
DS4. Основы вычислений			5						5	
DS5. Графы и деревья				4					4	
DS6. Дискретная вероятность			6						6	
PF1. Основные конструкции программирования	7	2							9	
PF2. Алгоритмы и решение задач	2	2		3					7	+1
PF3. Фундаментальные структуры данных	3	8		3					14	
PF4. Рекурсия	2	3							5	
PF5. Событийно-управляемое программирование		2				2		2	6	+2
AL1. Основы анализа алгоритмов		2		2					4	
AL2. Алгоритмические стратегии		2		6					8	+2
AL3. Фундаментальные вычислительные алгоритмы	3	3		6					12	
AL4. Распределенные алгоритмы						3			3	
AL5. Основы теории вычислимости	1			6					7	+1
AR1. Цифровая логика и цифровые системы			3		3				6	
AR2. Представление данных в памяти компьютера					3				3	
AR3. Организация машины на уровне ассемблера					9				9	
AR4. Устройство памяти компьютера					5				5	
AR5. Взаимодействие и коммуникации					3				3	
AR6. Функциональная организация					7				7	
AR7. Многопроцессорные и альтернативные архитектуры					3				3	
OS1. Обзор операционных систем						2			2	
OS2. Основы операционных систем						2			2	
OS3. Параллелизм						6			6	
OS4. Планирование и диспетчеризация						3			3	
OS5. Управление памятью						5			5	
NC1. Введение в распределенные вычисления						2			2	
NC2. Сети и телекоммуникации						7			7	
NC3. Сетевая безопасность						3			3	
NC4. Web как пример архитектуры "клиент-сервер"						3			3	
PL1. Обзор языков программирования		2							2	

PL2. Виртуальные машины		1							1	
PL3. Введение в трансляцию				2					2	
PL4. Переменные и типы данных	2	1							3	
PL5. Механизмы абстракции	1	2							3	
PL6. ОО-программирование	8	4				2			14	+4
HC1. Основы взаимодействия человека и машины		1					4	2	7	+1
HC2. Построение простого графического интерфейса								2	2	
GV1. Фундаментальные методы в графике	2							2	4	+2
GV2. Графические системы								1	1	
IS1. Основные вопросы интеллектуальных систем							1		1	
IS2. Поиск решений							5		5	
IS3. Представление знаний и вывод							4		4	
IM1. Информационные модели и системы							3		3	
IM2. Системы баз данных							3		3	
IM3. Моделирование данных							4		4	
SP1. История информатики	1								1	
SP2. Социальный контекст информатики								3	3	
SP3. Методы и средства анализа								2	2	
SP4. Профессиональная и этическая ответственность								3	3	
SP5. Риски и ответственность компьютерных систем	1							2	3	+1
SP6. Интеллектуальная собственность							3		3	
SP7. Конфиденциальность и гражданские свободы							2		2	
SE1. Проектирование ПО	2	2						4	8	
SE2. Использование программных интерфейсов приложений	1	1						3	5	
SE3. Программные средства и окружения	2							1	3	
SE4. Процессы разработки ПО								2	2	
SE5. Спецификации и требования к ПО		1						3	4	
SE6. Проверка соответствия ПО		1						2	3	
SE7. Эволюция ПО								3	3	
SE8. Управление программными проектами								3	3	
Сумма основных часов на каждый курс	**38**	**40**	**39**	**35**	**33**	**40**	**29**	**40**		

ГЛАВА 7. ВВОДНЫЕ КУРСЫ

Данная глава рассматривает вводную фазу базового учебного плана, во время которой студенты впервые сталкиваются с изучением информатики на университетском уровне. Раздел 7.1 описывает нашу общую философию относительно вводного учебного плана. Затем в разделах 7.2, 7.3 и 7.4 рассматриваются три основных вопроса, возникающих при проектировании вводных курсов: роль программирования, продолжительность вводного цикла и стратегии интегрирования в обучение дискретной математики. Раздел 7.5 перечисляет ряд концепций, знаний и навыков, которые, по нашему мнению должны быть частью идеальной вводной программы. Наконец, раздел 7.6 перечисляет шесть вводных стратегий, которые доказали свой успех на практике. В этом же разделе приведена критика каждого подхода, которая, как мы надеемся, поможет факультетам информатики при выборе варианта обучения, лучше всего отвечающего конкретным нуждам студентов, факультета, института и общества.

7.1. Общая философия

На протяжении всей истории компьютерного образования, структура курса по информатике была предметом горячих споров. За эти годы было предложено много стратегий, которые имеют как ярых сторонников, так и противников. Подобно проблеме выбора языка программирования, обсуждение стратегии построения вводных курсов по информатике слишком часто принимает характер религиозной войны, которая создает больше дыма, чем огня.

В целях примирения враждующих сторон, специальная комиссия CC2001 решила не рекомендовать ни одного подхода. Истина заключается в том, что еще не было найдено ни одной идеальной стратегии, и каждый подход имеет свои сильные и слабые стороны. Учитывая текущее положение дел в данной области, мы убеждены, что ни один универсальный подход не будет успешным во всех институтах. Так как вводные программы сильно различаются по своим целям, структуре, ресурсам и предполагаемой аудитории, то нужен ряд альтернативных стратегий, проверенных на практике. Более того, мы должны поощрять институты, факультеты и отдельных преподавателей продолжать эксперименты в этой области. В такой динамично меняющейся науке, как информатика, для повторения успеха необходимы постоянные педагогические нововведения.

7.2. Роль программирования во вводном цикле

Одним из наиболее горячо обсуждаемых вопросов в преподавании информатики является роль программирования во вводном учебном плане. На протяжении всей истории существования дисциплины большинство вводных компьютерных курсов фокусировались преимущественно на выработке навыков программирования. На распространенность "введения, ориентированного на программирование" (programming-first introduction) повлиял ряд практических и исторических факторов, включая следующее:

- Умение программировать является необходимым навыком для всех студентов, изучающих информатику. Овладение программированием в начале курса обучения гарантирует, что студенты имеют необходимые знания при переходе к основным и углубленным курсам.

- Информатика стала академической дисциплиной достаточно поздно, и к этому моменту в большинстве институтов уже появились вводные курсы по программированию, предназначенные для самого широкого круга слушателей. Поэтому к началу разработки учебных планов преподавания информатики вводные курсы уже однозначно воспринимались как "профессиональные курсы", ибо изначально программирование рассматривалось, в первую очередь, как инструмент. Таким образом, курсы по информатике зачастую строились как развитие существующих курсов по программированию, у которых никогда не было возможности развиться в более разностороннее введение в информатику как самостоятельную дисциплину.

- Модель "с ориентацией на программирование" была неявно поддержана ранними отчетами по учебным планам посредством структуры рекомендованных ими курсов. Например, *Рекомендации к учебным планам'68* [ACM68] начинаются с курса под названием "Введение в информатику", в котором подавляющее большинство тем связано с программированием. Центральная роль программирования во вводных курсах в дальнейшем была усилена определениями CS1 и CS2 в *Рекомендациях к учебным планам'78* [ACM78], которые определили эти курсы как цикл "Введение в программирование".

Однако, подход "с ориентацией на программирование" имеет несколько недостатков. Вот наиболее часто приводимые возражения против такого подхода:

- Концентрация на программировании за счет исключения других вопросов дает студентам ограниченное понимание дисциплины, укрепляя, таким образом, общее заблуждение, что "информатика тождественна программированию".

- Теоретические темы, которые должны укреплять понимание студентами практического материала, откладываются до более поздних этапов в обучении, когда они уже не имеют такой непосредственной значимости. Эта проблема имеет последствия как для студентов, специализирующихся в информатике, так и для студентов других специальностей. Студенты, прослушивающие только вводные курсы, лишаются понимания концептуальных основ, лежащих в основе революционных технологий, изменяющих наше общество. Что же касается студентов, специализирующихся в области информатики, тот факт, что теория не излагается в самых начальных курсах, укрепляет многих из них в необъективном мнении, что теория не имеет никакого отношения к их образовательным и профессиональным нуждам.

- Курсы по программированию зачастую сконцентрированы на синтаксисе и особенностях языка программирования. Это приводит к тому, что студенты больше концентрируются на этих относительно неважных деталях, чем на базовых алгоритмических навыках. В связи с этим многим студентам не удается осознать существование алгоритмической модели, выходящей далеко за рамки отдельных языков программирования. Более того, сосредоточенность на механистических деталях конструкций программирования чаще всего приводит к тому, что студенты вынуждены постигать содержательные моменты программирования методом проб и ошибок. Соответственно, такие курсы заставляют студентов, находящихся еще в самом начале своей академической карьеры, самостоятельно барахтаться в глубоком море программирования.

- Вводные курсы программирования зачастую чрезмерно упрощают процесс программирования, чтобы сделать его доступным для начинающих студентов. В результате, проектированию, анализу и тестированию уделяется недостаточно внимания, зато очень много внимания уделяется более простому процессу кодирования. Из-за этого студенты слоны переоценивать свои владение навыками программирования, что скрывает от них фундаментальные проблемы, которые будут мешать им в восприятии различных видов проблем и поиске решений.

- Курсы интенсивного обучения программированию ставят в невыгодное положение студентов, которые не имеют большого опыта в области компьютеров, создавая впечатление у тех, кто ранее пользовался компьютерами, что они знают больше, чем в действительности. В результате, студенты-новички в информатике зачастую просто не справляются с потоком информации, в то время как студенты с некоторыми предварительными знаниями чаще всего ограничиваются развитием своих плохих привычек.

- Подходы "с ориентацией на программирование" могут привести студентов к убеждению, что написание программы является единственным подходом к решению проблем с использованием компьютера. Однако в последнее время мощь и функциональность прикладных программ существенно увеличились, и студентам необходимо осознать, что подобные средства могут быть очень эффективными инструментами решения задач без использования классического программирования. Это соображение особенно важно для студентов, специализирующихся в других дисциплинах, так как их способности к решению задач и оценка собственных возможностей могут быть значительно улучшены путем обучения современным программным продуктам.

Несмотря на перечисленные недостатки, модель "с ориентацией на программирование" доказала свою исключительную жизнеспособность. Хотя в отчете CC1991 приводились доводы в пользу более широкого введения в дисциплину, большинство институтов продолжает концентрироваться во вводном цикле на программировании. Надо признать, что модель "с ориентацией на программирование" имеет ряд преимуществ, обуславливающих ее долговечность. Наиболее важными из этих преимуществ являются следующие:

- В большинстве институтов (по крайней мере, в США) основная масса слушателей вводного курса по информатике состоит из студентов, специализирующихся в других дисциплинах, но желающих овладеть навыками программирования. Факультеты, которые применяют стратегию "с ориентацией на программирование", могут использовать один и тот же курс как для тех, кто специализируется в компьютерной науке, так и для остальных студентов. Если же факультет использует альтернативную стратегию, которая полезна лишь для студентов, специализирующихся в информатике, – даже если предположить, что этот подход лучше! – то в дальнейшем может потребоваться разработка дополнительных курсов по программированию для удовлетворения потребностей других факультетов.

- Знание программирования являются необходимым условием для многих углубленных курсов по информатике. Учебные стратегии, откладывающие изложение основных навыков программирования, усложняют студентам возможность прослушивать столько углубленных курсов, сколько им, возможно, хотелось бы.

- Многим студентам программирование нравится больше, чем другие аспекты информатики. Поэтому курсы, базирующиеся на программировании, имеют тенденцию привлекать больше студентов к изучению информатики.

- Курсы по программированию дают навыки и тренинг, необходимые для студентов, их будущих работодателей, а также факультетов, не имеющих отношения к информатике.

Члены специальной комиссии CC2001 полагают, что модель "с ориентацией на программирование" будет оставаться доминирующей в обозримом будущем. Поэтому на специальную комиссию ложится обязанность предоставить рекомендации по оптимальному использованию этой модели. Мы осознаем, что на этом пути возникнут серьезные проблемы, и что для некоторых из этих проблем еще нет адекватного решения. Поэтому мы всячески поддерживаем нововведения и эксперименты с альтернативными моделями, пытающимися разрешить эти проблемы. Однако альтернативные подходы, которые пытаются оспорить доминирование модели "с ориентацией на программирование", должны принимать во внимание прагматические требования по выработке у прослушавших курсы студентов практических навыков.

В разделе 7.6 мы предлагаем три реализации модели "с ориентацией на программирование" и еще три подхода, базирующихся на альтернативных принципах. Реализации обучения с ориентацией на программирование – это традиционный подход с ориентацией на императивное программирование (imperative-first approach), подход с ориентацией на объектно-ориентированное программирование (objects-first approach), основанным на раннем изложении концепций объектов и объектно-ориентированного проектирования, а также подход с ориентацией на функциональное программирование (functional-first approach), представляющий алгоритмические концепции на примере языка с простым функциональным синтаксисом, например, Scheme. Во всех случаях мы пытались определить учебные модели, минимизирующие слабые стороны подхода "с ориентацией на программирование" посредством большей сосредоточенности на алгоритмических концепциях и методах решения задач, чем на причудах языкового синтаксиса. Три альтернативных модели – это подход с максимальным охватом материала (breadth-first approach), который начинается с общего обзора дисциплины, стратегия с ориентацией на алгоритмы (algorithm-first strategy), которая фокусируется на алгоритмах, а не на синтаксисе, и модель с ориентацией на аппаратную часть (hardware-first approach), которая начинается с электронных схем, а потом постепенно продвигается наверх по все усложняющимся уровням иерархии абстрактных машин.

7.3. Продолжительность вводного цикла

Хотя философия и структура вводных курсов сильно варьировалась на протяжении времени, один аспект обучения информатике остается на удивление постоянным: продолжительность вводного цикла. На протяжении нескольких десятилетий подавляющее большинство институтов использовало двухгодичный цикл для начального ознакомления студентов с информатикой. Среди специалистов, занимающихся вопросами преподавания информатики, эти два курса известны как CS1 и CS2 (названия заимствованы из *Рекомендаций к учебным планам'78* [ACM78]). Хотя содержание этих курсов с течением времени эволюционировало в связи с изменениями в технологиях и педагогическом подходе, продолжительность цикла оставалась неизменной.

Мы считаем, что сейчас как раз подходящий момент для пересмотра двухгодичного подхода к чтению вводных курсов. Количество и сложность тем, в которых должны разобраться поступающие студенты, существенно возросло. Задачи, встающие перед студентами, и инструменты, которые нужно использовать для их решения, стали намного сложнее. Все большее количество институтов обнаруживают, что двухгодичного вводного цикла теперь не хватает для того, чтобы охватить фундаментальные концепции программирования, особенно в тех случаях, когда эти вводные курсы пытаются предложить студентам более широкий обзор информатики. Расширение вводного цикла до трех курсов позволяет охватить все разрастающуюся совокупность знаний, в то же время предоставляя студентам достаточное количество времени на освоение материала.

Специальная комиссия CC2001 всячески поддерживает концепцию движения к вводному циклу, состоящему из трех курсов, и считает, что этот вариант окажется оптимальным для сравнительно широкого круга институтов. В то же время, вводный цикл из трех курсов подойдет не для всех институтов. Тот факт, что традиционный вводный цикл из двух курсов можно уместить в один год обучения в институтах с семестровой системой обучения, упрощает внедрение вводного материала без пересечений с материалом, предназначенным для второго курса обучения. Подобным же образом, проблема назначения зачетных баллов (credits) для курсов, прослушанных в других институтах (включая углубленные программы профессиональной подготовки в средних школах), становится более сложной, если один институт следует двухсеместровой системе, в то время как другие излагает вводный материал за три семестра.

Для того, что бы поддержать и двух- и трехкурсовые вводные циклы, специальная комиссия CC2001 описала обе альтернативы для трех из шести подходов к чтению вводных курсов, для которых трехкурсовая модель, скорее всего, будет иметь самые большие преимущества, а именно, "с ориентацией на императивное программирование", "с ориентацией на объектно-ориентированное программирование", и "с максимальным охватом материала". Подобные расширения могли бы быть разработаны и для трех других подходов, но они не включены в этот отчет. Для каждого из подходов, двух- и трехкурсовые варианты различаются номерами, присвоенными курсам. Трехкурсовый цикл обозначен цифрами 101, 102 и 103; двухкурсовый цикл обозначен цифрами 111 и 112.

7.4. Введение дискретной математики в начальный цикл обучения

Как обсуждается в главе 9, специальная комиссия CC2001 считает, что студентам, изучающим информатику, важно изучить дискретную математику как можно раньше, предпочтительно в первый год обучения. Существует как минимум две разумных стратегии для достижения этой цели:

1. Потребовать от студентов, изучающих информатику, прослушать курсы по дискретной математике одновременно с вводным циклом. Описания курсов в Приложении Б включают в себя две реализации курса по дискретной математике: семестровый курс (CS115), охватывающий основной объем материала из области Дискретных структур (DS) в совокупности знаний, и более полный, но более медленный вариант, разделенный на два курса (CS105 и CS106), в которых излагается весь требуемый материал, а также некоторые полезные дополнительные темы.

2. Интегрировать, как минимум, часть материала по дискретной математике непосредственно во вводный цикл информатики таким образом, чтобы студенты могли лучше понять, как эти математические инструменты применяются в практическом контексте. Хотя есть определенные выгоды в применении этого подхода при изложении некоторых тем, необходимо убедиться, что студенты в достаточной мере разбираются в дискретной математике для овладения преподаваемым материалом. Учитывая объем знаний в области дискретных структур (DS), оказывается невозможным объединить все требуемые темы во вводном цикле по информатике без добавления дополнительного математического курса. Поэтому типичные реализации будут включать некоторый материал непосредственно в цикл информатики, но при этом сохранят отдельный курс по дискретным структурам для того, что бы обеспечить необходимый охват материала. Реализация трех курсов подхода "с максимальным охватом материала" (CS101в/102в/103в) использует эту модель внедрения математических модулей непосредственно во вводные курсы по информатике.

7.5. Ожидания от вводного цикла

Несмотря на постоянные споры о полезности тех или иных педагогических подходов, существует множество принципов, разделяемых практически всеми преподавателями информатики. В этом разделе мы опишем наше понимание всеобщего консенсуса о минимальном наборе целей вводного учебного плана по информатике. Все стратегии, упомянутые в разделе 7.6, пытаются достичь много большего, чем мы описываем здесь, но каждая из этих стратегий охватывает общий набор тем, который может служить базой для дальнейшего изложения основных курсов.

В современном мире компьютеры используются повсеместно. Из-за важности компьютерных систем и широкой применимости навыков использования компьютера, начальный курс информатики определенно должен знакомить студентов с проектированием, конструированием и применением компьютерных систем, а также давать им навыки, доказавшие свою практическую полезность. В то же время вводные курсы по информатике должны знакомить студентов с главными интеллектуальными аспектами дисциплины. Когда мы рассматриваем информатику как дисциплину, нам нужно отвлечься от распространенного восприятия информатики как инструмента и сконцентрироваться на ее концептуальных основах. На каких принципах она базируется? Какие новые концепции привносит информатика в наш мир? Какого рода вопросы задают ученые, занимающиеся компьютерными системами? Какими подходами пользуются для решения проблем?

Мы считаем, что возможно разработать вводный цикл курсов по информатике, который решал бы все из перечисленных ниже задач:

- Знакомил студентов с набором фундаментальных концепций информатики
- Содействовал развитию когнитивных моделей для этих концепций
- Поощрял развитие у студентов навыков, необходимых для применения концептуальных знаний
- Упрощал перевод студентов-выпускников двухгодичных колледжей на четырехгодичные программы путем установления четких результатов обучения и определения соответствия между предлагаемыми учебными курсами

Для достижения этих целей на рисунке 7-1 перечислен ряд концепций, знаний и навыков, которые, по нашему мнению, должны стать частью каждого вводного учебного плана по информатике. Несмотря на то, что последовательность знакомства с темами и время, уделяемое различным темам, будут различными в разных учебных программах, мы ожидаем, что все вводные программы будут пытаться достигнуть поставленных целей. Рисунок 7-2 содержит схожие рекомендации в терминах разделов и тем из совокупности знаний по информатике, введенных в главе 5.

Рисунок 7-1. Концепции, заложенные в вводный учебный план

Алгоритмическое мышление		
Концепция	*Описание*	*Связанная деятельность*
Алгоритмы и вычисления	Алгоритмы как примеры вычислительного процесса; примеры важных алгоритмов	Читать и объяснять алгоритмы; обсуждать алгоритмическую корректность; использовать и адаптировать стандартные алгоритмы; писать алгоритмы
Эффективность алгоритмов и использование ресурсов	Простой анализ алгоритмической сложности; оценка анализа альтернатив; технические приемы оценки и измерений	Оценивать время и используемую память; проводить лабораторные эксперименты для оценки эффективности алгоритмов
Основы программирования		
Концепция	*Описание*	*Связанная деятельность*
Модели данных	Стандартные структуры для представления данных; абстрактные (представленные моделью) и конкретные (представленные реализацией) описания	Читать и объяснять значения программных объектов; создавать, использовать и изменять программы, манипулирующие стандартными структурами данных
Управляющие структуры	Результаты применения операций к программным объектам; что делает операция (описано моделью); как это делает операция (описано реализацией)	Читать и объяснять результаты операций; реализовывать и описывать операции; создавать программы для реализации ряда стандартных алгоритмов
Порядок исполнения	Стандартные управляющие структуры: последовательность, выборка, итерация; вызов функций и передача параметров	Применять надлежащим образом управляющие структуры при разработке алгоритмов и реализовывать эти структуры в программах
Инкапсуляция	Неразделимое объединение связанных объектов; взгляд со стороны клиента, основанный на абстракции и скрытии информации; взгляд со стороны реализации, основанный на внутренних деталях	Использовать существующие инкапсулированные компоненты в программах; проектировать, реализовывать и документировать инкапсулированные компоненты
Связи между инкапсулированными компонентами	Роль интерфейсов как связующего звена при обмене информацией; ответственность инкапсулированных компонентов перед их клиентами; роль наследования	Объяснять и применять наследование и интерфейсы; использовать наследование и интерфейсы при проектировании и реализации программ
Тестирование и отладка	Важность тестирования; стратегии отладки	Проектировать эффективные тесты; выявлять и исправлять логические и синтаксические ошибки
Вычислительные окружения		
Концепция	*Описание*	*Связанная деятельность*
Уровни абстракции	Компьютерные системы как иерархия виртуальных машин	Описывать роли различных уровней в иерархии виртуальных машин
Языки программирования и парадигмы	Роль языков программирования; процесс трансляции; существование разных парадигм программирования	Описывать процесс трансляции программы; определить как минимум две парадигмы программирования и описать их различия
Основные аппаратные средства и представление данных	Основы машинной организации; представление данных на машинном уровне	Объяснять элементы машинной структуры; показывать, как различные виды информации могут быть представлены при помощи битов
Инструменты	Компиляторы, редакторы, отладчики и другие компоненты программных окружений	Успешно использовать инструменты при разработке программного обеспечения
Приложения	Web-браузеры, текстовые процессоры, электронные таблицы, базы данных, системы электронной почты	Эффективно применять стандартные компьютерные приложения

Рисунок 7-2. Модули, охваченные всеми шестью вводными циклами

Разделы, в которых должны быть охвачены все темы:

DS1. Функции, отношения и множества
DS2. Основы логики
DS4. Основы вычислений
DS6. Дискретная вероятность
PF1. Основные конструкции программирования
PF4. Рекурсия
PL1. Обзор языков программирования
PL2. Виртуальные машины
PL4. Переменные и типы данных
PL5. Механизмы абстракции
SP1. История информатики

Разделы, в которых должна быть охвачена только часть тем

DS3. Методы доказательства: структура формальных доказательств; методы доказательства: прямой, с помощью контрпримера, методом противопоставления, через сведение к противоречию; математическая индукция

PF2. Алгоритмы и решение задач: стратегии решения задач; роль алгоритмов в процессе решения задач; концепция и свойства алгоритмов; стратегии отладки

PF3. Фундаментальные структуры данных: простые типы; массивы; записи; строки и обработка строк; представление данных в памяти; виды размещения в памяти: статическое, стековое и в "куче"; управление памятью во время выполнения программы; ссылки и указатели; связанные структуры

AL1. Основы анализа алгоритмов: нотация О-большого; стандартные классы сложности; эмпирические измерения эффективности; компромисс между временем и пространством в алгоритмах

AL3. Фундаментальные вычислительные алгоритмы: простые численные алгоритмы; алгоритмы последовательного и двоичного поиска; алгоритмы сортировки с квадратичной сложностью и со сложностью O(N log N); хеширование; деревья двоичного поиска

AR1. Цифровая логика и цифровые системы: логические вентили; логические выражения

PL6. Объектно-ориентированное программирование: объектно-ориентированное проектирование; инкапсуляция и скрытие информации; разделение поведения и реализации; классы, суперклассы, наследование; полиморфизм; иерархии классов

SE1. Проектирование ПО: фундаментальные концепции и принципы проектирования; объектно-ориентированный анализ и проектирование; проектирование с целью повторного использования

SE2. Использование программных интерфейсов приложения (API): программирование с использованием API; браузеры классов и дополнительные инструменты; программирование на примерах; отладка кода, использующего API

SE3. Программные средства и окружения: программные окружения; инструменты тестирования

SE5. Спецификации и требования к ПО: важность спецификации в программном процессе

SE6. Проверка соответствия ПО: основы тестирования; генерация тестовых примеров

7.6. Стратегии реализации вводного цикла

Этот раздел описывает шесть реализаций вводных курсов, которые специальная комиссия CC2001 оценивает как достигшие необходимого уровня успеха для признания их "лучшими практиками". Определение успешности подхода является намного более сложным процессом, чем это может показаться. Альберт Шанкер (Albert Shanker), бывший президент Американской федерации преподавателей (American Federation of Teachers), писал, что "образовательные эксперименты обречены на успех", хотя бы благодаря энергии и энтузиазму их авторов. При достаточном уровне энтузиазма практически любая педагогическая инновация будет успешной до тех пор, пока ее сторонники будут оставаться преданными своей идее. Настоящая же проверка состоит в том, удается ли воспроизвести успех при использовании данного подхода другими.

При выборе нашего ряда моделей для первого года обучения, мы настаивали на требовании, что подход должен быть поддержан людьми, которые с успехом использовали этот подход и при этом *не являются его создателями*. Таким образом, мы надеемся ограничить рассмотрение только теми стратегиями, которые уже имеют историю успешной реализации. Каждая из шести моделей, описанных в следующих разделах, удовлетворяет приведенному выше критерию успешной воспроизводимости. В то же время необходимо отметить, что этот критерий применялся только к общим подходам, но не к конкретным вариантам курсов, описанным в Приложении Б. В частности, нам не удалось найти существующих примеров реализации трехсеместровых вводных курсов, которые мы предложили в рамках подходов "с ориентацией на императивное программирование", "с ориентацией на ОО-программирование" и "с максимальным охватом материала". Сами по себе подходы доказали свою успешность при традиционной двухсеместровой организации, и мы считаем, что есть веские основания полагать, что трехсеместровые реализации достигнут такого же уровня успеха.

7.6.1. Подход с ориентацией на императивное программирование

Подход "с ориентацией на императивное программирование" (мы будем также использовать сокращенное название "императивный подход") – это самый традиционный из всех подходов, включенных в данный отчет. Как отмечено в разделе 7.3, мы предложили две реализации модели "с ориентацией на императивное программирование", одна из которых охватывает материал в три семестра, а вторая занимает два семестра, как показано ниже:

$CS101_I$. Основы программирования
$CS102_I$. Объектно-ориентированная парадигма
$CS103_I$. Структуры данных и алгоритмы

$CS111_I$. Введение в программирование
$CS112_I$. Абстракция данных

Двухсеместровая модель является более традиционной реализацией. $CS111_I$ предлагает введение в программирование в императивном стиле, используя структуру, подобную курсу CS1 (см. *Рекомендации к учебным планам'78* [ACM78, Koffman84]). Затем $CS112_I$ расширяет эти знания путем ознакомления студентов с большой частью материала традиционного курса CS2 [Koffman85], но с концентрацией на программировании в объектно-ориентированной парадигме. Трехсеместровая реализация ($CS101_I$/102_I/103_I) предлагает более широкий охват большинства тем, что позволит студентам лучше овладеть этими фундаментальными концепциями до того, как двигаться дальше. Кроме того, в такой реализации курсов предусмотрено время для дополнительных тем, которые дадут студентам более широкое видение дисциплины.

Важно отметить, что первый курс в обоих циклах – $CS101_I$ и $CS111_I$ – вполне может использовать объектно-ориентированный язык для примеров и упражнений по программированию. Основное отличие этого подхода от объектно-ориентированной модели заключается в акценте и порядке следования начальных тем. Даже если преподавание ведется с использованием объектно-ориентированного языка, первый курс фокусируется на императивных аспектах этого языка: выражениях, управляющих структурах, процедурах и функциях, а также других центральных элементах традиционной процедурной модели. Технологии объектно-ориентированного проектирования отложены на следующий курс.

"Императивный подход" сохраняет все недостатки – равно как и преимущества – любой реализации "с ориентацией на программирование" (подробное изложение см. выше в разделе 7.2). Кроме того, принятие "императивного подхода" означает, что студенты получат меньше опыта в технологии объектно-ориентированного программирования, чем это возможно при использовании модели "с ориентацией на ООП". Учитывая ключевую роль объектно-ориентированного программирования в требованиях к учебным планам, а также трудности, с которыми сталкиваются студенты при написании своих первых объектно-ориентированных программ, становится ясно, что откладывание этого материала на второй год обучения является слабостью описываемого подхода. В то же время, студентам действительно необходимо понимание традиционного императивного стиля программирования, который все еще широко используется и при этом является неотъемлемой частью любого объектно-ориентированного языка. Поэтому существуют различные мнения о том, какая модель должна быть представлена первой. Одни утверждают, что студенты, начинающие обучение с императивной модели, имеют больше проблем при переходе к объектно-ориентированному подходу. Другие возражают, что студенты, начинавшие с объектно-ориентированного языка, будут раздражены, если впоследствии им придется работать без тех конструкций, которые делают объектно-ориентированное программирование столь мощным.

В любом случае, институтам, применяющим "императивный подход", необходимо уделить дополнительное внимание ОО-темам в последующих курсах.

7.6.2. Подход с ориентацией на объектно-ориентированное программирование

Подход "с ориентацией на объектно-ориентированное программирование" (мы будем также говорить сокращенно "объектный подход") также фокусируется на программировании, но при этом с самого начала делает акцент на

принципах объектно-ориентированного проектирования и программирования. Как и в случае с императивной моделью, мы предлагаем как двух-, так и трехсеместровую реализацию, а именно:

$CS101_O$. Введение в объектно-ориентированное программирование
$CS102_O$. Объекты и абстракция данных
$CS103_O$. Алгоритмы и структуры данных

$CS111_O$. Объектно-ориентированное программирование
$CS112_O$. Объектно-ориентированное проектирование и методология

Первый курс в каждом цикле начинается непосредственно с понятия объектов и наследования. После экспериментов с этими идеями в контексте простых интерактивных программ, курс переходит к представлению более традиционных структур управляющей логики, но всегда в контексте общих понятий объектно-ориентированного проектирования. Последующие курсы более детально излагают материал по алгоритмам, основным структурам данных и вопросам программной инженерии.

Главным преимуществом "объектного подхода" к чтению вводных курсов является раннее ознакомление с объектно-ориентированным программированием, которое стало чрезвычайно важным как для академической среды, так и для промышленности. Например, в декабре 2000 г. совет колледжей (College Board) анонсировал свои планы по усилению объектно-ориентированного подхода в программе специализированных курсов (Advanced Placement, [AP2000]). В то же время объектно-ориентированная модель не избавлена от недостатков, общих для подходов "с ориентацией на программирование" (см. раздел 7.2). Фактически, проблемы подхода "с ориентацией на программирование" могут еще более остро проявиться в "объектном подходе", так как многие из языков, которые используются для объектно-ориентированного программирования в индустрии – в частности C++, а также в известной степени Java – намного сложнее классических языков. Если преподаватели не предпримут особых усилий по ограничению сложности в излагаемом материале, то детали выбранного языка могут подавить студентов, прослушивающих вводный курс.

7.6.3. Подход с ориентацией на функциональное программирование

Подход "с ориентацией на функциональное программирование" (или, сокращенно, "функциональный подход") был впервые использован в Массачусетском технологическом институте в 1980-х годах и характеризуется использованием в первом курсе простого функционального языка, такого как Scheme. В сравнении с другими реализациями, данный подход имеет следующие преимущества:

- Использование языка, редко используемого в промышленности, уменьшает эффект от разницы в подготовке поступивших студентов, часть из которых всегда имеет некоторый опыт в программировании.

- Минималистский синтаксис функциональных языков позволяет преподавателям фокусироваться на фундаментальных вопросах.

- Некоторые важные идеи – в особенности рекурсия, связанные структуры данных и использование функций как данных – совершено естественным образом вписываются в такой подход, и могут быть изложены в самом начале курса обучения.

Однако в этом подходе существуют и некоторые опасности. Первая опасность заключается в том, что студенты могут скептически относиться к изучению языка, который они потом никогда не будут использовать. Для студентов, которые уже выбрали информатику в качестве своей специальности, такие возражения можно преодолеть путем демонстрации выразительной силы используемого языка и задач, которые могут быть решены с помощью такого языка программирования. Студентам, прослушивающим вводный курс в качестве средства определения своей основной специальности, и особенно студентам, рассматривающим этот курс как способ получения некоторых практических навыков программирования, функциональные языки даются намного труднее. Вторая опасность заключается в том, что этот подход, как правило, требует от студентов более абстрактного мышления на ранней стадии обучения, чем это происходит при использовании более традиционных языков программирования. Заставлять студентов абстрактно мыслить определенно полезно, но всему должно быть свое время: столь ранний переход к абстракции может отвратить от информатики студентов, не привыкших к такому образу мышления.

Для изложения материала, необходимого для первого года обучения, вводный курс, следующий "функциональному подходу", должен быть продолжен интенсивным курсом, охватывающим объектно-ориентированное программирование и проектирование. Примерами курсов, которые реализовывают эту стратегию, являются:

$CS111_F$. Введение в функциональное программирование
$CS112_F$. Объекты и алгоритмы

7.6.4. Подход с максимальным охватом материала

В течение многих лет специалисты по образованию в области информатики беспокоилось о том, что традиционное введение "с ориентацией на программирование" дает студентам слишком ограниченный взгляд на дисциплину. Информатика является постоянно расширяющейся дисциплиной, которая включает в себя много других видов деятельности помимо программирования. Курсы, концентрирующиеся только на программировании, не позволяют студентам испытать многие другие области знаний и способы мышления, являющиеся частью современной информатики.

Для предоставления студентам более целостного взгляда на дисциплину, многие преподаватели предпочитают подход "с максимальным охватом материала", при котором на первом курсе рассматривается более широкий спектр тем (мы будем также сокращенно называть этот подход "обучение вширь"). Этот подход был настойчиво рекомендован как в отчете CC1991, так и в более раннем документе "Информатика как дисциплина", где утверждалось, что "первые курсы по информатике должны излагать не только программирование, алгоритмы и структуры данных, но также и материал из всех других поддисциплин" и что "математика и другие теоретические дисциплины должны быть интегрированы в программу обучения информатике" [Denning89].

Однако, разработка успешной реализации подхода "с максимальным охватом материала" оказалась трудной задачей. Наш опрос показал, что наиболее часто реализацией данной идеи было создание вводного обзорного курса, рассчитанного как на студентов, специализирующихся в области информатики, так и на остальных студентов. Такой курс дает студентам представление о целом ряде интересных и важных тем, вместо того, чтобы сразу погружать их в детали одной конкретной области. Студенты, заинтересовавшиеся той или иной областью, могут далее начать "обычный" одногодичный вводный цикл. Таким образом, в большинстве случаев речь идет о добавлении одного вводного курса "с максимальным охватом материала" в качестве отправной точки для всех студентов. Студенты, которые прошли подобный курс, могут двигаться дальше к любому другому вводному циклу, более подробно излагающему свой предмет.

Преимуществом использования курса "с максимальным охватом материала" в качестве вступления является то, что при таком подходе студенты могут сразу же оценить объем информатики, что позволит им быстрее определиться, хотят ли они более глубоко изучать данную область. В свою очередь, основным недостатком этого подхода является то, что он добавляет один курс к объему профилирующих куров и задерживает на один семестр завершение вводного цикла.

В результате обсуждений в рамках специальной комиссии CC2001 мы пришли к выводу, что нет никаких причин считать невозможным создание удачного цикла "с максимальным охватом материала", особенно, если отказаться от двухсеместрового ограничения на длительность вводного цикла. Основная заключается в перегруппировке материала традиционных курсов первого года (введение в программирование и дискретную математику) для предоставления студентам максимально широкого взгляда на область. К сожалению, мы не смогли найти такие модели, которые отвечали бы нашему критерию воспроизводимости успеха не только на факультете-создателе. Соответственно, мы представляем две отдельных реализации подхода "с максимальным охватом материала":

- Односеместровый курс (CS100в), который служит предпосылкой для более традиционного цикла по программированию
- Предварительная реализация вводного цикла "с максимальным охватом материала" (CS101в/102в/103в), которая пытается уместить в три семестра то, что (как показывает опыт) трудно уложить в два семестра

Другой подход к организации "максимального охвата" в учебном плане заключается в том, что бы предложить обзор дисциплины *после* завершения вводного цикла по программированию. Этот подход означает, что студенты начинают с "программистского" введения, в рамках которого они приобретают необходимые практические навыки, а затем им предоставляется возможность оценить разнообразие тем, являющихся частью информатики. Пока мы считаем, что подобный подход стоит опробовать в качестве эксперимента, поскольку мы еще не нашли моделей, которые отвечали бы нашим критериям приемлемости.

7.6.5. Подход с ориентацией на алгоритмы

В этом подходе основные концепции информатики представляются с использованием псевдокода вместо реального языка программирования. За счет ознакомления студентов с основными алгоритмическими концепциями и логическими структурами независимо от какого-либо языка программирования, этот подход минимизирует усилия, уходящие на изучения специфических синтаксических конструкций. Вместо этого, от студентов требуется обоснование и разъяснение алгоритмов, которые они создают, отлаживая их на бумаге и с помощью своего воображения. Это позволяет студентам работать с широким диапазоном типов данных и структур управляющей логики, без необходимости бороться с различными специфическими особенностями, которые неизбежно присутствуют в популярных языках программирования. Более того, так как студенты освобождены от необходимости исполнять свои программы на компьютерах, этот подход позволяет студентам ознакомиться с разнообразием таких конструкций значительно быстрее. Как только студенты овладевают основными алгоритмами и типами данных,

они могут начинать использовать один из распространенных языков программирования – или к концу первого семестра, или, самое позднее, в начале второго. Поскольку к этому моменту студенты уже знакомы с широким спектром структур данных и управляющих структур, традиционное программирование может быть пройдено намного быстрее, а аудиторное время может быть посвящено более глубокому изучению практических вопросов эффективного программирования и систематическому улучшению навыков отладки.

Благодаря исключению из программы обучения времени, отведенного на синтаксис и детали определенной среды программирования, вводный курс, следующий подходу "с ориентацией на алгоритмы", может включать в себя дополнительные теоретические темы, такие как оценка эффективности и основы вычислимости. Это может быть полезным в двух отношениях:

1. Студенты, специализирующиеся по другим дисциплинам, получают некоторое представление об информатике как "науке".

2. Студенты, специализирующиеся в информатике, начинают знакомиться с соответствующими аспектами теории с самых первых дней обучения, сокращая риск того, что под конец обучения они будут воспринимать теорию как неуместный "довесок" к учебной программе.

В то же время, подход "с ориентацией на алгоритмы" имеет несколько критических слабостей. Во-первых, первокурсники полны энтузиазма и всегда хотят заставить компьютер *сделать* что-то. Курсы, сводящиеся только к конструированию алгоритмов в псевдокоде, вызывают у таких студентов жестокое разочарование. Поэтому полезно объединить подход "с ориентацией на алгоритмы" с лабораторными работами, основанными на современных средствах разработки, и дающими студентам практический опыт в области информатики. Эта стратегия также помогает студентам развить набор практических навыков, который для студентов других дисциплин может быть даже более важным, чем традиционное программирование. При должной синхронизации программы лабораторных работ с лекционным материалом, студенты могут на собственном опыте ощутить важность, например, структур данных в контексте работы с базами данных, управляющих структур в контексте разработки электронных таблиц и высокоуровневого проектирования в контексте создания web-страниц.

Кроме того, ориентация на псевдокод также избавляет студентов от необходимости демонстрировать, что их алгоритмы работают и имеют функционирующую реализацию. Хотя процесс доведения программы до компилируемого и исполняемого состояния зачастую неприятен, овладение этим навыком критично для последующей успешной работы и потому студенты должны практиковаться в отладке на самой ранней стадии своего обучения. Процесс отладки псевдокода существенно отличается от отладки исполняемой программы. В первом случае мы говорим о "проверке за столом" и рассуждениях об алгоритме; во втором – как правило, об интерпретации симптомов и о "детективной" работе по нахождению программных ошибок. Оба навыка важны, и достаточно трудно определить, как подход "с ориентацией на алгоритмы" влияет на способность студентов отлаживать программы.

Последняя проблема в подходе "с ориентацией на алгоритмы" заключается в том, что он требует значительных усилий при выставлении оценок. Хотя заведомо неправильно оценивать программы только на основании результатов прохождения набора контрольных примеров, сама возможность подобных проверок обычно помогает преподавателям быстрее находить алгоритмические ошибки. В то же время оценка псевдокода на корректность – это значительно более трудная задача, которая, как правило, требует вовлечения большого количества ассистентов.

В этом отчете подход "с ориентацией на алгоритмы" реализуется следующими курсами:

$CS111_A$. Введение в алгоритмы и приложения
$CS112_A$. Методология программирования

Первый курс начинает с обсуждения алгоритмов и их приложений, а затем переходит к изложению основ объектно-ориентированного программирования и продолжает его до конца курса. Второй курс предлагает более тщательное изучение объектно-ориентированного программирования.

7.6.6. Подход с ориентацией на аппаратную часть

Подход "с ориентацией на аппаратную часть" (мы будем также называть его сокращенно "аппаратный подход") учит основам информатики, начиная с машинного уровня и продвигаясь затем к более абстрактным концепциям. Основная философия, стоящая за этой стратегией, утверждает, что таким образом студенты могут изучить информатику последовательно и без каких-либо мистификаций. Программа курса начинается с переключающих схем, которые затем используются для конструирования простых регистров и арифметических устройств, из которых, в свою очередь, строится простая фон-неймановская машина. Только после создания у студентов твердого понимания аппаратной части, курс переходит к рассмотрению программирования на языках высокого уровня.

Курсами, составляющими эту модель, являются:

CS111н. Введение в компьютер
CS112н. Технологии объектно-ориентированного программирования

Первый курс цикла в мельчайших подробностях излагает устройство компьютера; второй курс использует этот базис для того, чтобы развить у студентов навыков программирования и дать солидное введение в объектно-ориентированные технологии.

Такой подход хорошо работает для студентов, предпочитающих понимать процесс вычисления во всех его деталях. При этом он менее эффективен для того, чтобы поощрять студентов видеть целостные концепции, стоящие за механизмом реализации. "Аппаратный подход" также не очень хорошо согласуется с растущей централизацией всех процессов вокруг программного обеспечения и современной тенденцией все большего совершенствования виртуальных машин, отделяющих процесс программирования от лежащих в основе технических средств. В то же время, мы считаем, что подобный подход может быть вполне жизнеспособным в программах по проектированию компьютеров (computer engineering), в которых необходимо добиться раннего ознакомления с техническими средствами.

ГЛАВА 8. ОСНОВНЫЕ КУРСЫ

Основные курсы в учебном плане должны дать студентам серьезный базис для последующего глубокого изучения отдельных тем. В то же время необходимо помнить, что вводные курсы, описанные в главе 7, и основные курсы, описанные здесь, *не* представляют собой полный курс обучения. Все университетские программы будут включать в себя значительное количество дополнительного материала по выбору (описание этих курсов приведено в главе 9). В данной главе мы предлагаем четыре варианта реализации учебного плана преподавания основного уровня:

- Традиционный подход, в котором отдельные курсы посвящены самостоятельным темам
- Сокращенный подход, который организовывает курсы вокруг более общих тем
- Интенсивный системно-ориентированный подход
- Подход с ориентацией на WWW, использующий сеть в качестве основного лейтмотива

Эти варианты реализации представляют собой скорее показательные модели, чем предписанные стандарты. В каждом случае есть много других работоспособных вариантов, использующих схожие философии для разработки несколько отличающихся наборов курсов. Более того, зачастую есть возможность использовать гибридные подходы, комбинируя элементы из разных моделей, как описано в разделе 8.3. Главный вопрос ко всем этим вариантам достаточно прост: гарантирует ли рассматриваемая реализация, что все студенты ознакомятся со всеми обязательными разделами знаний за время своего обучения.

8.1. Проблемы при составлении основных курсов

Как и в случае с вводными курсами, различные факультеты и институты разработали множество подходов к построению курсов основного уровня. Это разнообразие и педагогическое экспериментирование, сделавшее его возможным, являются хорошими знаками, свидетельствующими о жизненности образования в нашей дисциплине. Однако члены специальной комиссии CC 2001 обеспокоены тем, что большинство существующих моделей представляют собой набор относительно обособленных тем и лишь немногие пытаются построить учебный план вокруг абстрактных тем, объединяющих всю дисциплину. В 1992 году, в статье под названием "Мы можем обучать информатике лучше" [Shaw92], Мери Шоу выразила это соображение следующим образом:

> *Давайте организуем наши курсы вокруг идей, а не вокруг артефактов. Это сделает наши цели более ясными, как для студентов, так и для преподавателей. Машиностроительные институты не преподают проектирование бойлера – они преподают термодинамику. В то же время, как минимум два из основных курсов по информатике – "Создание компиляторов" и "Операционные системы" – являются артефактными динозаврами программирования.*

Мы считаем, что этот совет применим и сегодня. В существующей программе обучения информатике, многие курсы продолжают концентрироваться на артефактах. Мы считаем, что составителям программ обучения и преподавателям необходимо оторваться от этого наследия прошлого и начать экспериментировать с альтернативными моделями.

Многие предыдущие отчеты пытались определить общие темы, объединяющие практику информатики. Например, в CC1991 определено три общих процесса и двенадцать концепций, которые пронизывают обучение информатике (см. рис. 8-1). Аналогично, модель учебного плана ACM для гуманитарной степени по информатике (liberal arts degree in computer science) [Gibbs86, Walker96] организована путем разделения всех основных курсов на четыре центральные темы информатики (технические средства, программное обеспечение, алгоритмы и теория), с соответствующими курсами для каждой темы.

Как минимум, мы считаем, что даже базирующиеся на артефактах подходы к обучению должны меняться по мере изменения технологий. Как заметила Мери Шоу, традиционный курс обучения включает в себя курсы по операционным системам и компиляторам, которые уже не так важны для нашей дисциплины как раньше. В современном мире сети и компонентные системы являются более значимыми на практике. В такой динамично развивающейся сфере, как информатика, специализация на конкретных приложениях, производителях либо реализациях делает знания студентов уязвимыми и быстро устаревающими.

Рисунок 8-1. Процессы и темы из отчета CC1991

Три процесса	*Двенадцать концепций*
Теория: • Определения и аксиомы • Теоремы • Доказательства • Определения и аксиомы **Абстракция:** • Сбор данных и построение гипотезы • Моделирование и прогнозирование • Планирование эксперимента • Анализ результатов **Проектирование:** • Требования • Спецификации • Проектирование и реализация • Тестирование и анализ	• Связывание • Сложность больших проблем • Концептуальные и формальные модели • Целостность и завершенность • Эффективность • Эволюция • Уровни абстракции • Упорядочение в пространстве • Упорядочение во времени • Повторное использование • Обеспечение безопасности • Компромиссы и их последствия

8.2. Примеры подходов для основного уровня

Последующие подразделы описывают четыре основных подхода, выделенных в этом отчете, а именно тематический, сокращенный, системно-ориентированный и WWW-ориентированный. Кроме того, в дополнительном разделе мы описываем смешанный подход. Для понимания того, какие курсы уровня 200 применяются совместно, в описаниях используются индексы T, C, S и W, чтобы курс мог быть увязан со своей моделью. Например, для ссылки на основной курс по архитектуре при обсуждении тематического подхода используется обозначение $CS220_T$. "Сжатый" курс обозначается как $CS220_C$. На самом деле, эти два курса – одно и то же. Для основных курсов номер курса однозначно определяет материал, а индексы используются только для указания, какая модель (или модели) включают в себя этот курс. В описании курсов в Приложении Б, в заголовке указываются все модели, которые используют данный курс. Например, заголовок

$CS220_{\{C,S,T\}}$. Архитектура ЭВМ

показывает, что этот курс используется в сокращенном, системном и тематическом подходах.

8.2.1. Традиционный тематический подход

Наиболее типичным подходом к чтению основных курсов является простая группировка материала, основанная на традиционном разделении области. Таким образом, студенты прослушивают отдельные курсы по каждой из основных областей: курс по архитектуре, курс по операционным системам, курс по алгоритмам и т.д. Однако необязательно читать отдельные курсы для каждой из областей, входящей в обязательный список. Некоторые области с относительно небольшим количеством обязательных модулей (например, графика), могут быть интегрированы во вводную программу обучения. Другие области, такие как человеко-машинное взаимодействие, могут быть объединены с углубленными курсами, рассматривающими природу профессиональной практики дисциплины.

В качестве варианта реализации этой модели мы предлагаем следующий набор курсов:

$CS210_T$. Разработка и анализ алгоритмов
$CS220_T$. Архитектура ЭВМ
$CS225_T$. Операционные системы
$CS230_T$. Распределенные вычисления
$CS260_T$. Искусственный интеллект
$CS270_T$. Базы данных
$CS280_T$. Социальные и профессиональные вопросы
$CS290_T$. Разработка программного обеспечения
CS490. Курсовой проект

Эта модель очень близка к используемой в данный момент на многих факультетах, поэтому она проверена временем. Курсы лекций, скорее всего, будут не новы для большинства преподавателей, а необходимые преподавательские ресурсы – такие как учебные планы, тексты и наборы задач – легко доступны.

В то же время, эта модель подвержена всем проблем "курсов, основанных на артефактах", описанным в разделе 8.1. К тому же, для некоторых институтов размер этой модели может быть проблемой. Студенты должны прослушать девять основных учебных курсов в дополнение к одной из вводных последовательностей, описанных в главе 7. Для больших учебных заведений девять основных курсов вряд ли будут проблемой. Однако для институ-

тов с ограниченными ресурсами, поддержка девяти основных курсов может оказаться невозможной. Сокращенный подход, описанный в следующем разделе, предлагает способ уменьшения размера учебного плана, как, впрочем, и смешанные подходы, описанные в разделе 8.3.

8.2.2. Сокращенный подход

Итак, тематический подход посвящает полный учебный курс каждой из основных областей в совокупности знаний. Однако большинство из этих областей не включают в себя 40 часов обязательного материала, и это означает, что в тематическом подходе основные курсы включают в себя не только обязательные, но и дополнительные темы. Для институтов, которым необходимо сократить количество основных курсов, самым лучшим подходом будет объединение отдельных тем в *тематические курсы*, собирающие материал связанных между собой областей информатики в единое целое. Помимо того, что эта стратегия уменьшает количество читаемых курсов, она также помогает в борьбе с проблемой курсов обучения, основанных на "программных артефактах".

Например, определенно можно объединить материал по искусственному интеллекту из $CS260_T$ и материал по базам данных из $CS270_T$ в один интегрированный курс, рассматривающий обе эти идеи. Аналогично, можно объединить обязательные темы по программной инженерии с темами из социальной и профессиональной области. Наш сокращенный подход представляет собой результат чрезвычайно агрессивной попытки объединения тем, что позволило сократить вводный цикл традиционной модели с девяти курсов до пяти:

$CS210_C$. Разработка и анализ алгоритмов
$CS220_C$. Архитектура ЭВМ
$CS226_C$. Операционные системы и сети
$CS262_C$. Управление информацией и знаниями
$CS292_C$. Разработка ПО и профессиональная практика

Эта реализация охватывает все обязательные разделы знаний, но экономит четыре курса по сравнению с тематическим подходом. В результате, эта модель может быть полезна в тех случаях, когда необходимо придерживаться минимального количества основных курсов. Такое сжатие приемлемо для маленьких колледжей с небольшим количеством преподавателей на факультете или для любого института, оказавшегося не в состоянии нанять на работу достаточное количество преподавателей для чтения более объемных программ.

Конечно, необязательно полностью придерживаться тех вариантов сжатия основных курсов, которые используются в данной модели. Раздел 8.3 описывает ряд смешанных подходов, которые перенимают стратегию сокращенного подхода для создания циклов основных курсов с размерами от пяти курсов сжатой модели до девяти курсов традиционной модели. В то же время важно не переусердствовать с сокращением количества курсов. Мы крайне не рекомендуем пытаться сгруппировать все обязательные модули в теоретический минимум из семи курсов, основываясь на том, что ядро состоит из 280 часов. Излишнее уплотнение программы обучения создает курсы, в которых темы недостаточно логически связаны и которые оставляют слишком мало времени и свободы преподавателям для адаптации и улучшения материала.

8.2.3. Системно-ориентированный подход

В конечном счете, теория и практика информатики находят свое применение в разработке высококачественных компьютерных систем. Этот раздел определяет учебный план обучения информатике, использующий разработку систем в качестве объединяющей темы. Такая модель включает в себя больше технического и профессионального материала, чем другие модели, при сохранении рационального уровня охвата теоретических вопросов. Теория информатики остается важной как в качестве основы для понимания практики, так и в качестве надежного фундамента знаний, остающихся актуальными, несмотря на постоянные изменения технологий.

Минимальная реализация системно-ориентированного подхода состоит из следующих курсов, продолжающих вводный цикл:

CS120. Введение в устройство компьютера
$CS210_S$. Разработка и анализ алгоритмов
$CS220_S$. Архитектура ЭВМ
$CS226_S$. Операционные системы и сети
$CS240_S$. Трансляция языков программирования
$CS255_S$. Компьютерная графика
$CS260_S$. Искусственный интеллект
$CS271_S$. Управление информацией
$CS291_S$. Разработка ПО и системное программирование
CS490. Курсовой проект

Хотя названия этих курсов наводят на мысль, что они концентрируются на отдельных областях, необходимо использовать всеобъемлющее понятие системы в качестве обобщающей темы. Эта системная перспектива должна

пронизывать все аспекты программы обучения и включать некоторую комбинацию теории, практики, приложений и соответствующего подхода.

8.2.4. WWW-ориентированный подход

Эта модель выросла из массового спроса на циклы учебных курсов, которые уделяют больше внимания Интернету и World Wide Web, используя эти области в качестве общего базиса для программы обучения в целом. Следующие курсы представляют одну из попыток разработки подобной модели:

CS130. Введение в WWW
$CS210_W$. Разработка и анализ алгоритмов
$CS221_W$. Архитектура ЭВМ и операционные системы
$CS222_W$. Архитектура сетей и коммуникаций
$CS230_W$. Распределенные вычисления
$CS250_W$. Человеко-машинное взаимодействие
$CS255_W$. Компьютерная графика
$CS261_W$. Искусственный интеллект и информация
$CS292_W$. Разработка ПО и профессиональная практика

8.3. Смешанные подходы

Как было отмечено во введении к этой главе, четыре основных подхода, выделенных в этом отчете – традиционный, сокращенный, системно-ориентированный и WWW-ориентированный – должны рассматриваться как показательные модели, которые демонстрируют только некоторые из доступных возможностей. Во многих случаях разумно объединить элементы двух или более подходов для создания нового смешанного учебного плана, который может более эффективно отвечать требованиям отдельного института. При таком объединении необходимо удостовериться, что получившийся в результате комбинирования учебный план охватывает все обязательные темы.

Рисунок 8-2 показывает три смешанных подхода, охватывающих все обязательные темы при использовании в сочетании с любым из вводных циклов, описанных в главе 7. Возможны и другие комбинации.

Все подходы, описанные в данной главе (т.е. четыре конкретных модели и различные смешанные типы), имеют общую цель: представить основные идеи и устоявшиеся концепции информатики, которые должен изучить каждый студент для успешной работы в этой области. Выполняя эту задачу, основные курсы закладывают основу для более углубленной работы в области информатики.

Рисунок 8-2. Смешанные подходы

Смешанный подход из восьми курсов, добавляющий один комплексный курс к традиционной модели:
$CS210_T$. Разработка и анализ алгоритмов
$CS220_T$. Архитектура ЭВМ
$CS226_C$. Операционные системы и сети
$CS260_T$. Искусственный интеллект
$CS270_T$. Базы данных
$CS280_T$. Социальные и профессиональные вопросы
$CS290_C$. Разработка ПО
CS490. Курсовой проект

Смешанный подход из семи курсов, который соединяет WWW и сокращенный подходы:
CS130. Введение в WWW
$CS210_W$. Разработка и анализ алгоритмов
$CS221_W$. Архитектура и операционные системы
$CS222_W$. Операционные системы
$CS230_W$. Распределенные вычисления
$CS262_C$. Управление информацией и знаниями
$CS292_C$. Разработка ПО и профессиональная практика

Смешанный подход из шести курсов, который соединяет традиционный и сокращенный подходы:
$CS210_T$. Разработка и анализ алгоритмов
$CS220_T$. Архитектура ЭВМ
$CS225_T$. Операционные системы
$CS230_T$. Распределенные вычисления
$CS262_C$. Управление информацией и знаниями
$CS292_C$. Разработка ПО и профессиональная практика

ГЛАВА 9. ЗАВЕРШЕНИЕ УЧЕБНОГО ПЛАНА

Основной целью глав 7 и 8 было формулирование различных подходов к обучению обязательным разделам информатики. Как мы уже отмечали в этом отчете, обязательные для изучения разделы информатики ("ядро") сами по себе не представляют полного курса обучения. Для завершения учебного плана необходимо, чтобы студенты получили разнообразные знания и навыки, а также получили возможность выполнить сложную работу, выходящую за пределы минимального объема. Данная глава предлагает стратегии и направления для достижения всех этих результатов. Раздел 9.1 описывает набор общих требований к образованию студентов в области информатики. Раздел 9.2 предлагает набор углубленных учебных курсов. Затем в разделе 9.3 обсуждаются курсовые проекты. Наконец, раздел 9.4 содержит обзор нескольких моделей учебных курсов, ориентированных на различные типы институтов.

9.1. Общие требования

Успешному выпускнику факультета информатики необходимо множество навыков помимо чисто технических знаний по информатике. Например, студент, специализирующийся в области информатики, должен иметь определенный уровень математической подготовки, знать научные методы работы, понимать, представлять способы использования вычислительных методов на практике, обладать эффективными навыками общения, а также способностью продуктивно работать в коллективе. Данная глава содержит несколько общих рекомендаций по достижению перечисленных целей.

9.1.1. Математическая строгость

Математические методы и формальные рассуждения является составной частью большинства областей информатики. В отчете CC1991 теория определяется как один из трех основных базисов информатики, и мы абсолютно уверены, что этот принцип не устарел и сегодня. Информатика зависит от математики и ее фундаментальных определений, аксиом, теорем и методов доказательств. Кроме того, математика предоставляет язык для работы с понятиями, относящимися к информатике, конкретными средствами анализа и верификациями, а также в качестве теоретической основы для понимания важных идей информатики. Например, функциональное программирование и решение проблем основано на математических концепциях и нотациях для функций; анализ алгоритмов напрямую зависит от таких разделов математики, как комбинаторика и теория вероятностей; анализ параллелизма и предотвращение блокировок требует использования теории графов; наконец, верификация программ и анализ вычислимости базируются на формальной логике и дедукции. Таким образом, в программы по обучению информатике необходимо включать достаточный объем математики, чтобы студенты понимали теоретические основы дисциплины.

Учитывая глубокую роль математики в информатике, программы обучения должны включать математические концепции как можно раньше и как можно чаще. Основные математические концепции должны быть представлены студентам в начале обучения и на более поздних курсах эти концепции должны регулярно использоваться. Хотя колледжам и университетам придется адаптировать свои требования к предварительным знаниям абитуриентов для того, чтобы отразить локальные потребности и возможности, крайне важно использовать на старших курсах математические навыки, развитые на ранних курсах. Кроме того, сведения о предварительных требованиях к студентам необходимо зафиксировать в формальных документах факультета

При разработке этих предложений мы пришли к выводу, что программы по информатике должны взять на себя ответственность по включению необходимых разделов математики, особенно дискретной математики. Для этого данный отчет определяет новую область знаний, состоящую из дискретной математики, обязательной для университетской программы. Эта область – Дискретные Структуры (DS) – определяет темы и разделы, которые, по нашему мнению, особенно важны для основного курса. Материал по DS может быть введен как отдельный курс или интегрирован в процесс обучения путем совместного представления вместе с зависящими от него материалами по информатике. В любом случае, важно придавать особое значение методам дискретной математики на протяжении всего основного курса.

Специальная комиссия CC2001 дает следующие рекомендации относительно математического содержания в курсе информатики:

- *Дискретная математика*. Все студенты должны быть ознакомлены с приемами дискретной математики. При возможности, желательно дать более одного курса в этой области; в любом случае, все программы по информатике должны освещать, как минимум, все обязательные темы DS. Стратегии по интеграции дискретной математики во вводный цикл обсуждаются в разделе 7.4.

- *Дополнительная математика.* Студенты должны получать дополнительные знания по математике для развития своего мастерства в этой области. Эта математика может состоять из самых различных курсов, включая статистику, математический анализ, линейную алгебру, численные методы, теорию чисел, геометрию или ло-

гику. Выбор должен зависеть от целей программы обучения, потребностей учебного заведения и нужд самих студентов.

9.1.2. Научный подход

Как указано в отчете CC1991, процесс абстрагирования (сбор данных, построение и проверка гипотез, экспериментирование и анализ) представляет компоненту логического мышления, используемого в информатике. Научный подход является базовой методологией для большей части дисциплины информатики и студенты должны иметь ясное понимание этой методологии.

Чтобы в достаточной мере развить понимание научного метода, студенты должны приобрести собственный опыт в области формулирования гипотез, проектирования экспериментов, тестирования гипотез и анализа данных. Хотя учебный план может вырабатывать этот опыт у студентов различными способами, необходимо, чтобы студенты действительно "занимались наукой", а не просто "читали о науке".

В связи с этими соображениями специальная комиссия CC2001 дает следующие рекомендации о научной методологии:

- Студенты должны развивать свое понимание научных методов и получать научный опыт в курсах, включающих некоторый объем лабораторных работ.

- Студенты могут получать свой научный опыт в различных предметных областях, в зависимости от целей обучения и личных интересов.

9.1.3. Знакомство с приложениями

Учитывая широкую сферу применения информатики в современном обществе, специалисты по информатике должны уметь работать с людьми из других дисциплин. В связи с этим специальная комиссия CC2001 рекомендует всем специалистам по информатике:

- Принять участие в углубленном изучении какого-либо предмета, полагающегося на информатику для решения своих специфических задач.

Студенты в области информатики имеют широкий круг возможных интересов и профессиональных целей. Для многих из них изучение информатики совместно с некоторой предметной областью будет чрезвычайно полезно. Этого можно достичь несколькими путями. Один из способов заключается в интеграции в курсы по информатике практических примеров, подчеркивающих важность понимания предметной области. Другой вариант – это включение в программу обучения длительной стажировки или семестровой практической работы, которая зачитывалась бы в университетской программе обучения. Такие возможности уже определенно существуют в таких областях, как психология, социология, экономика, биология, бизнес и других научных и прикладных дисциплинах. При наличии некоторого воображения, можно найти приложения и в более далеких от информатики областях. Зачастую этого удается достичь с помощью новаторских методов, выходящих за рамки стандартных курсов по информатике.

9.1.4. Навыки общения

Распространенным требованием среди потенциальных работодателей является способность специалистов по информатике эффективно общаться с коллегами по работе и клиентами. Так как подобные навыки важны для любой карьеры в области информатики, студенты должны оттачивать свои навыки устной и письменной речи в различных ситуациях – как в рамках курсов по информатике, так и в других контекстах. В частности, студенты должны уметь:

- Эффективно выражать свои идеи в письменном виде.
- Делать эффективные устные презентации – как в формальной, так и в неформальной обстановках.
- Понимать и конструктивно обсуждать выступления других

Хотя учебные заведения могут следовать различным стратегиям для достижения этих целей, программа каждого студента должна включать множество возможностей для улучшения своих письменных и ораторских навыков, включая навыки устного изложения и восприятия устной речи.

Как минимум, учебный план по информатике должен включать в себя:

- Курсовую работу для развития письменных навыков**.**
- Как минимум, одну формальную устную презентацию в группе.
- Критическое обсуждение, по крайней мере, одного выступления.

Кроме того, учебный план по информатике должен содержательным образом включать в себя постоянные письменные и устные дискуссии. Навыки общения не должны рассматриваться отдельно, а, наоборот, должны быть интегрированы в учебный план и его требования.

9.1.5. Работа в коллективе

Лишь немногие профессионалы в области информатики могут рассчитывать на работу в изоляции долгое время. Программные проекты обычно выполняются группой людей, работающих как одна команда. Следовательно, студенты должны изучить механизм и динамику эффективного участия в коллективе в процессе своего университетского обучения. Более того, так как значение работы в коллективе (а также возникающих при этом трудностей) не становится очевидным в проектах небольшого масштаба, студенты должны принимать участие в командных проектах, рассчитанных на довольно длительный промежуток времени, скажем, на целый семестр или существенную его часть.

Для того чтобы студенты имели возможность приобрести эти знания в рамках университетских программ, специальная комиссия CC2001 рекомендует всем программам обучения включить в себя:

- Возможность работы в коллективе на ранних этапах обучения.
- Значительный проект, включающий в себя сложную задачу, проектирование и реализацию решения которой берет на себя небольшой студенческий коллектив. Этот проект чаще всего планируется на последний год обучения в университете и представляет собой дипломный проект. Стратегии структурирования подобной работы рассматриваются в разделе 9.3 этой главы.

Опыт, получаемый студентами в этом групповом проекте, может быть усилен путем использования групп студентов различных специальностей. Например, студенты-информатики могут сотрудничать со студентами-биологами в проекте, посвященном биовычислениям. Такой проект потребует знаний из обеих дисциплин, и стратегии эффективного междисциплинарного общения получат особое значение. Отчет ABET 2000 [ABET2000] особенно поддерживает концепцию междисциплинарных групповых проектов и наша специальная комиссия подтверждает, что подобные проекты могут дать особенно ценный опыт для студентов, как в области информатики, так и за ее пределами.

9.1.6. Дополнительный учебный план

В наше время, когда наблюдается активный спрос на специалистов в области информатики, учебные заведения испытывают сильное давление со стороны работодателей, желающих видеть у выпускников какие-либо конкретные навыки. С одной стороны, стремление выпускать профессионалов с необходимыми для рынка труда навыками, несомненно, является положительной. С другой стороны, необходимо иметь в виду, что для студентов наилучшая подготовка заключается не в овладении специфическими навыками, которые могут быстро устареть, а в получении прочных теоретических и практических знаний, которые позволят им продолжительное время оставаться на современном уровне. Лучше всего данный аспект студенческой подготовки может быть изложен следующим образом: работодатели и сами студенты должны рассматривать выпускников в области информатики как агентов преобразования, способных приносить с собой на работу навыки и ожидания, которые будут долго оставаться ценными для организации.

Чтобы направить студентов на этот путь, учебный курс должен содействовать развитию набора универсальных умений, повышающих эффективность выпускников. В некоторой степени эти умения включают в себя перечисленные в предыдущих разделах. Однако они включают в себя и некоторые качества, которые редко развиваются с помощью занятий, например, способность написания эффективного резюме, управление своим временем, проведение исследований в библиотеке, развитие профессиональной ответственности, способность оставаться в курсе современных исследований, развитие навыков постоянного самообразования и т.п. Эта группа умений определена как *дополнительный учебный план.*

Один из способов гарантировать развитие перечисленных навыков у студентов – это включение этих элементов в традиционный процесс обучения. Однако существует опасность того, что элементы дополнительного учебного плана заберут слишком много времени, необходимого для овладения техническим материалом. Разработчики учебного плана и отдельных курсов должны найти в этом тонком вопросе необходимый баланс.

9.2. Углубленные курсы

Термин *углубленные курсы* используется для обозначения курсов, содержание которых выходит за рамки обязательного материала ("ядра"). Разделы совокупности знаний свидетельствуют о богатом наборе возможностей, которые существуют для подобных курсов, но лишь немногие учебные заведения способны предложить набор курсов, детально покрывающих каждый модуль. Очевидно, организации будут стремиться ориентировать подобные курсы на свои собственные области компетенции, определяемые потребностями студентов, знаниями преподавателей и нуждами общества.

Одна из рабочих групп по педагогике, созданных специальной комиссией CC2001, создала набор углубленных курсов, основанных на описании совокупности знаний. Список названий этих курсов приводится на рисунке 9-1. Однако мы решили не включать в печатную версию отчета полное описание углубленных курсов, за исключением некоторых курсов, являющихся частью одного из учебных планов, описанных в главе 8. Вместо этого, планируется создание web-страниц для этих курсов, которые будут доступны с основного сайта CC2001 (http://www.computer.org/education/cc2001). Таким образом, мы рассчитываем сократить размер печатного документа и в то же время добиться большей актуальности описаний углубленных курсов.

Рисунок 9-1. Темы углубленных курсов по дисциплинам

Дискретные структуры (DS)
CS301. Комбинаторика
CS302. Вероятность и статистика
CS303. Теория кодирования и информации

Методы вычислений (CN)
CS304. Методы вычислений
CS305. Численный анализ
CS306. Исследование операций
CS307. Статистическое моделирование
CS308. Математическое программирование
CS309. Вычислительная биология

Алгоритмы и теория сложности (AL)
CS310. Анализ алгоритмов - 2
CS311. Теория языков и автоматов
CS312. Криптография
CS313. Геометрические алгоритмы
CS314. Параллельные алгоритмы

Архитектура и организация ЭВМ (AR)
CS320. Углубленные вопросы архитектуры компьютеров
CS321. Параллельные архитектуры
CS322. Однокристальные системы
CS323. Разработка VLSI
CS324. Кодизайн

Операционные системы (OS)
CS325. Операционные системы (2)
CS326. Параллельные и распределенные системы
CS327. Надежные вычисления
CS328. Отказоустойчивость
CS329. Системы реального времени

Распределенные вычисления (NC)
CS330. Углубленные вопросы компьютерных сетей
CS331. Распределенные вычисления
CS332. Программирование мобильных устройств
CS333. Кластерное программирование
CS334. Сжатие данных
CS335. Управление сетями
CS336. Сетевая безопасность
CS337. Корпоративные сети

Языки программирования (PL)
CS340. Создание компиляторов
CS341. Проектирование языков программирования
CS342. Семантики языков программирования
CS343. Парадигмы программирования
CS344. Функциональное программирование
CS345. Логическое программирование
CS346. Языки сценариев

Взаимодействие человека и машины (HC)
CS350. Проектирование и оценка эргономичных структур
CS351. Графические пользовательские интерфейсы
CS352. Разработка мультимедийных систем
CS353. Разработка диалоговых систем
CS354. Использование компьютеров в коллективе

Компьютерная графика и визуализация (GV)
CS355. Компьютерная графика (2)
CS356. Компьютерная мультипликация
CS357. Визуализация
CS358. Виртуальная реальность
CS359. Генетические алгоритмы

Интеллектуальные системы (IS)
CS360. Интеллектуальные системы
CS361. Автоматическое доказательство теорем
CS362. Системы с базами знаний
CS363. Обучение машины
CS364. Системы планирования
CS365. Обработка естественного языка
CS366. Агенты
CS367. Робототехника
CS368. Символьные вычисления
CS369. Генетические алгоритмы

Управление информацией (IM)
CS370. Базы данных (2)
CS371. Устройство баз данных
CS372. Обработка транзакций
CS373. Распределенные и объектные базы данных
CS374. Информационная проходка
CS375. Организация информационных хранилищ
CS376. Мультимедийные информационные системы
CS377. Электронные библиотеки

Социальные и профессиональные вопросы (SP)
CS380. Профессиональное программирование
CS381. Социальный контекст программирования
CS382. Компьютеры и мораль
CS383. Экономические вопросы программирования
CS384. Правовые вопросы, связанные с компьютерами
CS385. Интеллектуальная собственность
CS386. Право на частную жизнь и гражданские свободы

Программная инженерия (SE)
CS390. Разработка ПО (2)
CS391. Программная инженерия
CS392. Проектирование ПО
CS393. Проектирование ПО и формальные спецификации
CS394. Практическая программная инженерия
CS395. Улучшение процессов разработки ПО
CS396. Компонентное программирование
CS397. Программные окружения
CS398. Системы с повышенными требованиями к надежности

9.3. Курсы-проекты

Как обсуждалось в разделе 9.1.5, специальная комиссия CC2001 полагает, что для студентов весьма важно поучаствовать в крупном групповом проекте, который включает в себя и проектирование и реализацию. В зависимости от структуры университета, существует несколько стратегий для предоставления студентам такого практического опыта. В некоторых случаях возможна организация стажировки в местных компаниях, во время которой студенты могут реализовать свой проект в промышленной обстановке. Однако значительно чаще, факультеты информатики вынуждены встраивать такие практические проекты в структуру учебной программы.

В приложении Б предлагается несколько моделей по включению в учебный курс работы над проектами. Первой стратегией является включение части проекта в основной или углубленный курс по программной инженерии. Эта стратегия отражена в курсе

CS292$_{\{C,W\}}$. Разработка ПО и профессиональная практика,

который включает в себя коллективный проект, а также большое количество дополнительного материала. Если студентам предоставлено достаточно времени на разработку и осуществление серьезного проекта, то этот подход работает. Однако проекты в таких курсах обычно весьма невелики по масштабу, хотя бы потому, что время, отводимое на изучение материала по программной инженерии, сокращает время проекта.

В качестве альтернативы, специальная комиссия CC2001 рекомендует включать в учебный план большой заключительный проект, который позволит студентам применить все свои навыки и знания, полученные в течение периода обучения. Соответствующий курс может включать небольшое количество дополнительного материала, но основное внимание должно быть направлен на сам проект. Приложение Б включает как односеместровый, так и двухсеместровый варианты:

CS490. Основной проект

или

CS491. Основной проект I
CS492. Основной проект II

Двухсеместровая версия предлагает студентам намного больше времени на полное завершение проекта, но ее не всегда возможно уместить во временные ограничения учебных программ в США.

9.4. Примеры учебных планов

Одной из самых больших сложностей в разработке рекомендаций к учебным планам является огромное количество вариантов учебных программ, существующих в различных университетах и колледжах. Учитывая различие ожиданий от профессиональной подготовки (особенно при рассмотрении образования в разных странах, но также и в пределах США), невозможно создать одну простую модель, которая бы удовлетворяла все учебные заведения. Главы 7 и 8 предлагают несколько различных подходов к вводным и основным уровням обучения, которые могут быть адаптированы большинством институтов. Основная задача этого раздела – показать, как полный учебный план может быть интегрирован в программы обучения различных учебных заведений.

Возможно, наиболее значительное отличие между академическими программами – это количество курсов информатики, необходимых для завершения университетской программы обучения. В учебных заведениях за пределами США студенты университетов обычно концентрируются на одном предмете, возможно с некоторыми дополнительными курсами в тематически близких к основному предмету областях. При таком типе обучения студенты прослушивают 3-4 курса по информатике в первый год, 4-5 во второй год и по 5-6 в третий и четвертый годы обучения. В подобных учебных заведениях студенты должны закончить 17-21 курсов информатики при четырехлетнем обучении. В США такой уровень концентрации встречается чрезвычайно редко. Например, в университетах студенты обычно получают 12-15 курсов по информатике, а оставшаяся часть учебного плана заполняется обязательными общеобразовательными курсами и курсами по выбору. В гуманитарных колледжах студенты прослушивают 9-12 курсов по информатике, а остальное время заполняется большим количеством гуманитарных курсов и, зачастую, второй основной или дополнительной специальностью (second major or minor) в другой области изучения. Таким образом, количество курсов информатики, входящих в образовательную программу, может отличаться в два раза и более.

Очень важно понимать, что "небольшая программа обучения" *не* означает "слабая программа обучения". Любой учебный план, опирающийся на рекомендации, предложенные в этом отчетом, может обеспечить полноценное обучение основам информатики. Независимо от характеристик и ожиданий конкретного учебного заведения, любой учебный план должен:

- Покрывать все 280 часов обязательного материала из совокупности знаний по информатике.
- Требовать от студентов посещения углубленных курсов, как минимум, в одной области информатики.
- Включать соответствующий уровень математики.
- Предоставлять студентам возможность получения профессиональных навыков и знаний, необходимых в "реальном мире" (исследовательский опыт, коллективная работа, написание технических документов, работа над проектом и т.д.).

Следующие три раздела описывают модели учебных планов, разработанные для следующих основных типов учебных заведений:

1. Университеты США, ориентированные на исследовательскую работу.
2. Университеты, в которых образование сконцентрировано на изучении одного основного предмета, что типично для стран за пределами Северной Америки.
3. Учебные заведения, подобные гуманитарным колледжам в Соединенных Штатах, с небольшим факультетом информатики.

9.4.1. Модель учебного плана для исследовательских университетов в Соединенных Штатах

Цель данной модели заключается в демонстрации соответствия между CC2001 и типичными программами обучения американских университетов, ориентированных на исследовательскую работу. Такие университеты обычно имеют довольно большой штат преподавателей, позволяющих изучать информатику с достаточной степенью глубины и ширины охвата. В таких случаях подразумеваемой целью обычно является обучение всех студентов на уровне, достаточном для последующей профессиональной или научной карьеры. Другой целью для многих из университетов является предоставление возможности продолжения образования выпускникам двухгодичных образовательных колледжей (community colleges).

При разработке учебного плана университета, можно воспользоваться любой комбинацией вводных курсов, описанных в главе 7, с традиционным, системно-ориентированным или WWW-ориентированном подходом к чтению основных курсов (см. главу 8). Наиболее распространенной вводной последовательностью курсов в таких ситуациях являются двух- и трехкурсовые циклы "с ориентацией на императивное программирование" и "с ориентацией на ООП", описанные в разделах 7.6.1 и 7.6.2. Однако необходимо отметить, что реализация таких учебных программ не должна сводиться к фиксации существующей практики. Каждый из перечисленных вводных циклов привносит во вводные и основные курсы заметное количество современного материала, в особенности в таких областях как сети и базы данных. На сегодняшний день во многих университетах этот материал преподается только на углубленных факультативных курсах, которые могут быть попущены студентами.

Рисунок 9-2. Университетская модель (США)

	Первый семестр	Второй семестр
год 1	$CS101_I$. Основы программирования Вычисления 1	$CS102_I$. ООП CS115. Дискретные Вычисления 2
год 2	$CS103_I$. Структуры данных и алгоритмы Научный курс 1	CS120. Введение в организацию компьютеров Научный курс 2 Теория вероятности и статистика
год 3	$CS210_T$. Проектирование и анализ алгоритмов $CS220_T$. Архитектура ЭВМ Углубленный математический курс по выбору	CS225t. Операционные системы CS280t. Социальные и профессиональные аспекты CS. по выбору Исследовательский проект
год 4	$CS230_T$. Распределенные вычисления $CS262_T$. Управление информацией и знаниями $CS290_T$. Разработка ПО Исследовательский проект	CS490. Заключительный проект CS. по выбору CS. по выбору

Рисунок 9-2 отражает структуру учебного курса, разработанного для исследовательских университетов США. Следующие разделы предлагают дополнительные соображения по поводу решений, затрагивающих общую структуру модели.

Вводные и основные курсы

Как было отмечено в предыдущей части, любой вводный цикл совместно с любым основным циклом (за исключением сжатого подхода), является подходящим для исследовательских университетов. Например, учебный план, приведенный на рисунке 9-2, использует трехсеместровый "императивный" вводный цикл и традиционный подход к изложению курсов основного уровня, со следующими модификациями:

- Добавлен факультативный курс CS120 "Введение в организацию компьютеров" – этот вопрос обсуждался при описании системно-ориентированного подхода.

- Пара "традиционных" курсов по искусственному интеллекту и базам данных, CS260 и CS270, была замещена комбинированным курсом CS262 "Управление информацией и знаниями" – этот вопрос рассматривался при обсуждении сокращенного подхода.

CS120 является факультативным, поскольку все его обязательные модули покрываются также и другими курсами. Если CS120 включается в состав учебного плана, то в последующих курсах можно рассмотреть некоторые вопросы более глубоко. Если же этот курс не включается в учебный план, то он может быть заменен другим CS факультативом или просто исключен. В нашем примере CS120 упомянут по той причине, что университетские программы часто хотят добиться раннего и более глубокого покрытия системных материалов по сравнению с традиционным подходом. Кроме того, CS120 – это курс, который может преподаваться в двухгодичных колледжах.

Одним из сильных сторон CC2001 является обязательность материала по управлению данными и интеллектуальным системам. Многие учебные заведения выносят почти весь этот материал в факультативные курсы. В данном учебном плане мы предлагаем один обязательный курс, излагающий обе эти темы, в надежде, что учебные заведения будут дополнительно предлагать студентам факультативы по обоим курсам.

Научная методология и математика

Глубокие знания в науке и математике обычно являются одной из целей программ по информатике исследовательских университетов. Соответственно, мы рекомендуем посвятить два семестра науке. Учитывая наши требования о получении студентами глубоких и зрелых математических знаний, мы закладываем в данную программу изучение следующих курсов:

- Один семестр на изучение дискретных структур (см. курс "Дискретные структуры в информатике"). Институты, желающие предложить более глубокое изучение этого материала, могут расширить программу путем использования двухсеместровой последовательности "Дискретные структуры 1" и "Дискретные структуры 2".

- Введение в математический анализ на уровне, необходимом для посещения факультативных курсов по математике, таких как логика, линейная алгебра и абстрактная алгебра. В зависимости от учебного заведения, требования к курсу вычислений могут колебаться от одного до трех курсов и более. Мы полагаем, что для студентов, специализирующихся в информатике, предпочтительнее потратить меньше времени на математический анализ, но больше на дискретную математику или другие темы, более востребованные практикой информатики. Однако, во многих учебных заведениях факультеты информатики не могут повлиять на структуру программы по математике, что не позволяет факультету добиться достаточной гибкости в этом вопросе.

- Один семестр теории вероятностей и статистики.

- По крайней мере, один семестр углубленного факультатива по математике.

Окончание учебного плана

Для завершения своего образования, студенты должны изучить некоторый дополнительный материал, выходящий за рамки обязательного минимума. Во многих институтах имеет смысл оставить вопрос об области специализации на усмотрение самих студентов. Таким образом, один из путей формирования учебного плана является простое требование к студентам включить некоторое количество факультативных курсов в свою программу. Однако, для достижения более точных образовательных целей, факультеты могут регулировать свои требования. Например, программа может обеспечивать широкий охват материала путем требования факультативов из различных областей. И наоборот, программа может добиться большей глубины в какой-то области, если обязать студентов прослушать по ней несколько спецкурсов. В зависимости от локальных специализаций и интересов, некоторые институты могут явным образом требовать прослушивания одного или нескольких углубленных курсов. Например, институты, обращающие особое внимание на математический фундамент студентов, могут включить в свою обязательную программу курс CS310 "Углубленный алгоритмический анализ".

В дополнение к углубленному материалу, программа обучения должна знакомить студентов с проблемами программирования больших систем. Стратегии осуществления этого требования могут широко варьироваться. Студенты могут получить опыт программирования больших систем различными путями. Это может быть односеместровый курсовой/дипломный проект (CS490) или проект, рассчитанный на два семестра (CS491-CS492) или углубленный курс разработки программного обеспечения (CS390).

Одним из больших преимуществ исследовательских университетов для студентов является активное участие преподавателей в процессе расширения границ дисциплины. Однако для многих студентов этот аспект остается незаметным, потому что сравнительно небольшое количество студентов получают возможность участия в исследовательских проектах во время учебы. Студенты, которые все-таки получают такую возможность, приобретают целый ряд преимуществ:

- Интересный опыт творческого исследования.
- Связи с преподавателями, которые могут выступать в качестве студенческих наставников.

- Опыт работы над проектами, который может быть очень полезным как для будущей работы по специальности, так и для продолжения обучения в аспирантуре.

9.4.2. Предметно-ориентированная модель

Большинство курсов, прослушиваемых студентами университетов в Соединенных Штатах и Канаде, обычно находится *за пределами* их основной специализации. В других странах такой обобщающий подход является редким. Вместо этого, ожидается, что студенты будут концентрироваться на какой-то одной области, возможно дополненной несколькими курсами из близких дисциплин. Мы называем учебный план такого типа *предметно-ориентированным.* Такой метод типичен, например, для Англии, где учебные программы рассчитаны на три года. Другие страны обычно используют четырехгодичную модель, но в любом случае предметно-ориентированную модель достаточно легко адаптировать к локальным условиям.

Предметно-ориентированная модель предлагает некоторую степень гибкости на любом уровне обучения. Например, в первый год обучения эта гибкость может выражаться в возможности расширить свой взгляд на информатику путем выбора из большого количества факультативных курсов. Эти факультативы могут, например, освещать какие-либо интересные прикладные области и таким образом расширять и улучшать общую подготовку студентов. Кроме того, факультативные курсы помогают студентам проверить, уверены ли они в правильности своего выбора дисциплины. Детали этого процесса будут варьироваться в разных институтах и, скорее всего, будут зависеть от различных соображений, в частности, от уровня требований к поступающим студентам. Например, некоторые учебные заведения могут требовать от кандидатов определенных знаний по математике или даже по самой информатике. В таких случаях детали программ должны изменяться соответствующим образом.

Другая возможность воспользоваться гибкостью программы появляется в последний год обучения, когда программа факультативных углубленных курсов позволяет студентам выбрать свою специализацию, зачастую с целью изучения или развития своих карьерных возможностей в данной области. К этому моменту курсы уже должны подводить студентов к переднему краю изучаемой области (как минимум, с точки зрения высшего образования).

На рисунке 9-3 представлена предметно-ориентированная модель, рассчитанная на три года. В дополнение к основным рекомендациям данного отчета, в этот метод были заложены следующие решения:

- *Программирование – это трудный для преподавания предмет, который требует внимания и значительного времени в учебной программе.* Курсы, которые дают студентам базовые знания по программированию, являются крайне важными. Студенты должны иметь возможность как можно чаще практиковать свои навыки в программировании, чтобы более поздние курсы могли базироваться на опыте, полученном в ранних курсах.

- *Программа должна предоставлять студентам широкие возможности для развития практических навыков.* Большинство курсов в программах по информатике должно включать лабораторные работы; это позволит студентам развивать свои технические навыки и приобрести опыт и понимание эффективных профессиональных приемов. Для успешной аттестации по курсу студенты должны продемонстрировать соответствующий уровень мастерства в данной практической области.

- *Модельная программа не содержит конкретного курса по научным методам; вместо этого предполагается, что соответствующий материал может быть интегрирован в структуру факультативных курсов.* Например, метод эксперимента может быть изучен в курсе "Человеко-машинное взаимодействие". Обучение подобному материалу в контексте вопросов информатики значительно предпочтительнее, чем самостоятельное изложение в отрыве от основного материала

- *По возможности, желательно преподавать вспомогательный материал в контексте его применимости к информатике.* Замечание о необходимости преподавания материалов с учетом контекста применимо к разработке любого учебного плана. Большая часть вспомогательного материала, включая математику, может быть более эффективно представлено студентам в контексте его применения.

Рисунок 9-3. Предметно-ориентированная модель.

	Первый семестр	**Второй семестр**
год 1	$CS101_O$. Введение в ООП CS105. Дискретные структуры 1 CS120. Введение в организацию компьютеров	$CS102_O$. Объекты и абстракция данных CS106. Дискретные структуры 2 Вероятность и статистика

год 2	$CS103_O$. Алгоритмы и структуры данных $CS210_S$. Разработка и анализ алгоритмов $CS220_S$. Архитектура ЭВМ $CS271_S$. Управление информацией	$CS226_S$. Операционные системы и сети $CS240_S$. Трансляция языков программирования $CS255_S$. Компьютерная графика $CS291_S$. Разработка ПО и системное программирование
год 3	$CS260_S$. Искусственный интеллект CS380. Профессиональная практика CS. по выбору CS491. Заключительный проект 1	CS326. Параллельные и распределенные системы CS393. Программная инженерия и формальные спецификации CS. по выбору CS492. Заключительный проект 2

9.4.3. Модель для малых факультетов

Эта модель разработана для программ по информатике, реализуемых небольшими факультетами. Слово "небольшой" используется неформально, так как факультет, небольшой по меркам одних университетов, может быть "весьма большим" в других. В основном эта модель ориентирована на факультеты с количеством преподавателей не более пяти или шести, но, тем не менее, может быть использована и для более крупных факультетов.

Основным последствием небольшого количества преподавателей на разработку учебного плана является то, что количество курсов информатики будет меньшим, чем у больших факультетов. Например, модель университета, описанная в разделе 9.4.1, содержит 15 курсов по информатике, предметно-ориентированная модель из раздела 9.4.2 содержит 21 курс. Чтение такого количества курсов в учебном заведении с пятью-шестью преподавателями информатики не представляется возможным. Специализация в подобных учебных организациях обычно включает 9-11 курсов по информатике, а также вспомогательные курсы по математике и заключительную проектную работу.

Модель для малого факультета проиллюстрирована на рисунке 9-4. Она определяется 14 курсами, организованными в следующие группы:

1.	Вспомогательные математические курсы	3
2.	Вводные курсы по информатике	2
3.	Основной цикл курсов по информатике	5
4.	Углубленные факультативы по информатике	3
5.	Заключительный проект	1
	Всего курсов	14

Курсы каждой из этих групп детально описаны ниже.

Вспомогательные математические курсы

Количество вспомогательных курсов часто определяется доступным объемом времени в учебном плане. Несмотря на то, что желательно иметь четыре или пять подобных курсов, зачастую невозможно обеспечить подобный уровень математической подготовки без ущерба для других требуемых программой курсов. Мы рекомендуем следующий *минимум* обязательной математики (с оговоркой, что при наличии времени дополнительные курсы математики крайне желательны):

CS105. Дискретные структуры 1.
CS106. Дискретные структуры 2.
Как минимум один дополнительный курс математики по выбору студента, а также другие углубленные факультативы, используемые для завершения программы.

Рисунок 9-4. Модель для малых факультетов.

	семестр 1	*семестр 2*
год1	CS111o. Объектно-ориентированное программирование CS105.Дискретные структуры 1	CS112o. Объектно-ориентированные дизайн и методология CS106. Дискретные системы 2
год2	CS210c. Проектирование и анализ алгоритмов CS220c. Архитектура ЭВМ	CS226c. Операционные системы и сети Математический факультатив
год3	CS262c. Управление информацией и знаниями CS. факультатив	CS292c. Разработка ПО и профессиональная практика CS. факультатив
4	CS. факультатив	CS490. Заключительный проект

Мы выбрали двухсеместровый подход к изучению дискретной математики, потому что темы этого курса являются наиболее важной частью математики для специалистов в области информатики. Сегодня большинство институтов предлагает курс, рассчитанный на один семестр. Однако в данной области так много обязательного материала, что два семестра позволяют создать значительно более эффективный курс.

Третий математический курс обязательной программы не был определен. Он должен быть выбран совместно с научным руководителем, основываясь на интересах студента и тех дополнительных курсах, которые они собираются выбирать. Это может быть линейная алгебра, математическая логика, математическое моделирование, численный анализ и т.п.

Вводные курсы по информатике

Для вводных курсов рекомендуется любая из следующих двухсеместровых последовательностей курсов, определенных в главе 7:

$CS111_O$. Объектно-ориентированное программирование
$CS112_O$. ОО проектирование и методология

или

$CS111_F$. Введение в функциональное программирование
$CS112_F$. Объекты и алгоритмы

Оба из этих вводных циклов концентрируются на важных концептуальных идеях, таких как решение проблем, разработка спецификаций и языковые парадигмы, а не на синтаксических деталях конкретных языков программирования. Эти циклы хорошо подходят для небольших факультетов, так как они знакомят студентов со многими фундаментальными идеями и концепциями при небольшом количестве курсов.

Основные курсы информатики

Для набора основных курсов мы выбрали сокращенный подход, представленный в разделе 8.2.2. Эта модель содержит пять обязательных курсов, которые покрывают все 280 часов обязательного материала. Вот эти пять курсов:

CS210c. Проектирование и анализ алгоритмов.
CS220c. Архитектура компьютеров.
CS226c. Операционные системы и сети.
CS262c. Управление информацией и знаниями.
CS292c. Разработка ПО и профессиональная практика.

Есть несколько причин, по которым этот набор курсов предпочтителен для небольших факультетов. Во-первых, обязательная программа сжата всего лишь в пять курсов. Это позволяет покрыть все 280 часов обязательных модулей небольшим количеством учебных курсов, которые могут читаться даже самыми небольшими факультетами. Второй, и даже более важной, причиной является отличие этих курсов от традиционной, "базирующейся на артефактах", программы, подразумевающей наличие отдельных курсов по всем "материальным" темам программирования, таким как компиляторы, операционные системы, базы данных, сети, графика и WWW. Вместо этого данная модель включает набор *комплексных* (cross-cutting) курсов, каждый из которых объединяет родственный материал из различных областей информатики. Например, $CS262_C$ "Управление информацией и знаниями" интегрирует материал по СУБД и искусственному интеллекту с такими алгоритмическими темами, как сжатие данных и кодирование. Наконец, темы этики и профессионализма не выносятся при таком подходе в отдельный курс, независимый от остальной программы. Взамен, этот важный материал был включен во многие основные курсы.

Например, $CS262_C$ рассматривает вопросы интеллектуальной собственности, а $CS292_C$ "Разработка ПО и профессиональная практика" включает в себя темы социального контекста информатики, этической и профессиональной ответственности, а также темы риска и ответственности при разработке программного обеспечения.

Углубленные факультативы информатики

Углубленные курсы преследуют следующие три цели:

1. Изложение углубленных вопросов, выходящих за пределы обязательного материала.
2. Демонстрация приложений фундаментальных концепций, представленных в основном курсе.
3. Глубокое изучение, по крайней мере, одной области информатики.

Аналогично математическим курсам, точное число факультативов по информатике будет зависеть от наличия свободных мест в учебном плане после компоновки основных курсов. Тем не менее, число факультативов должно быть достаточно большим, чтобы предоставить студентам возможность углубленного изучения, по крайней мере, одной области информатики. Мы предлагаем минимум из трех углубленных факультативов, хотя и осознаем, что некоторые институты могут увеличить или уменьшить это число, исходя из местных условий. Мы полагаем, что три факультативных курса могут дать достаточную возможность для углубления знаний, в то же время удерживая размеры полной программы в управляемых пределах.

Для получения достаточно глубоких знаний, по крайней мере, в одной области информатики, имеет смысл сделать обязательным прослушивание студентами двух факультативов из трех в данной области знаний. Разбиение углубленных курсов по областям показано на рисунке 9-1.

Заключительный проект

Последней компонентой этой модели является курс CS490, заключительный проект. Этот курс дает студентам возможность отточить те навыки и умения, которые не могут быть развиты при традиционном обучении в классе, например, работа в коллективе, взаимодействие с пользователями, формальная спецификация задач, просмотр научных журналов, создание прототипов, написание научных статей и искусство устных презентаций.

Наиболее популярной моделью такого проекта является коллективная разработка программного обеспечения, в которой студенты проектируют решение информационной проблемы и работают в командах для реализации этих решений. Однако существует и другая модель, которая может показаться привлекательной для студентов, желающих в дальнейшем поступить в аспирантуру и заниматься исследованиями, а не работать в промышленности. Для этих студентов интересной альтернативой будет опыт самостоятельной исследовательской деятельности, который включает в себя некоторую подготовительную работу, изучение научной литературы и исследование предложенного решения с последующей подготовкой научной статьи и/или устной презентации результатов. Нельзя забывать, что это всего лишь студенческая работа и потому необходимо реалистично оценивать объем и качество такой исследовательской деятельности. Но даже с учетом этого соображения, для выдающихся студентов столкновение с трудностями исследований может оказаться значительно полезнее, чем разработка еще одной программы.

Наконец, каждый институт должен определить сроки выполнения заключительного проекта. Оптимальным сроком (особенно для исследовательского проекта) представляется один год. Однако недостаток ресурсов на малых факультетах может привести к выполнению заключительного проекта за один семестр.

9.4.4. Программа для колледжей с двухгодичным курсом обучения

В США многие студенты начинают свое обучение в области информатики не с четырехлетних программ институтов, а с двухлетних программ колледжей. В результате, программы обучения информатике в таких колледжах являются критически важными для отчета CC2001. Так как двухлетние программы колледжей имеют свои собственные ограничения и соображения, которые могут несколько отличаться от четырехлетних программ, специальная комиссия CC2001 в сотрудничестве с комиссией по двухлетним колледжам ACM и его недавно созданным аналогом в IEEE-CS решила опубликовать отдельный отчет, предлагающий более конкретные рекомендации для двухлетних колледжей.

Несмотря на существование отдельного отчета для двухлетних колледжей, некоторые аспекты соответствующей модели важны и для четырехгодичных институтов. Центральным вопросом, связывающим двух- и четырехгодичные учебные заведения является вопрос *перевода*, то есть определение возможности для выпускников колледжей эффективно продолжить свое обучение по четырехгодичной программе для получения законченного высшего образования. Этот вопрос очень важен для четырехлетних институтов, принимающих выпускников двухлетних колледжей, и требует дальнейшего обсуждения в данном отчете.

Программа двухлетних колледжей обычно попадает в одну из двух категорий – ориентированная на дальнейшую карьеру или на дальнейшее обучение – в зависимости от типа учебного заведения и потребностей местной инду-

стрии. Программа, ориентированная на карьеру, обычно дает образование, ограниченное конкретными знаниями, навыками и способностями, необходимыми для дальнейшей непосредственной работы. Студенты, окончившие двухлетнюю "карьерную" программу, обычно сразу приступают к работе. После получения некоторого опыта профессиональной деятельности некоторые выпускники карьерно-ориентированных программ, могут поступить в четырехлетние институты для завершения своего образования; другие выпускники могут сделать это сразу. С другой стороны, в программе, ориентированной на дальнейшее обучение, ожидается, что большинство студентов продолжит свое образование в четырехлетних институтах. Если двухлетняя программа обучения изначально не ориентирована на такой переход, то студентам, возможно, придется прослушать дополнительные курсы на вводном цикле или в начале основного цикла.

Тщательное и внимательное согласование тем и курсов двух- и четырехлетних программ облегчает переход студентов из одних учебных заведений в другие. Основной задачей согласования программ является сглаживание трудностей такого перехода. Для эффективного согласования требуется тщательная оценка курсов и программ, а также осмысленное общение и сотрудничество. Однако, этот процесс довольно сложен по нескольким причинам:

1. Слушатели двухлетних программ часто приходят извне традиционной студенческой среды и их предыдущие опыт и подготовка могут различаться сильнее, чем у студентов четырехлетних институтов.

2. Так как многие двухлетние колледжи предлагают учебные программы, связанные со стажировкой, знания студентов этих колледжей представляют собой смесь теории и практических навыков, которую трудно отобразить на традиционную четырехлетнюю программу.

3. В двух- и четырехлетней программах трудно отыскать курсы, соответствующие один-к-одному. Более вероятно нахождение аналогичных последовательностей курсов, хотя количество курсов в этих последовательностях может быть различным.

В связи с этими трудностями, необходимо, чтобы все учебные заведения воспринимали согласование программ как постоянный процесс достижения и поддержания договоренностей.

Преподаватели учебных заведений обоих типов должны обеспечить четкое описание учебных программ, жесткое следование указанным в программах целям и адекватную оценку студентов в соответствии с описанными стандартами знаний. Если учебные заведения подписывают договор о согласовании программ, факультеты двухгодичных колледжей должны обеспечить преподавание материала, необходимого для подготовки студентов к дальнейшей академической работе, в не меньшем объеме, чем для того же материала в четырехгодичных программах.

Программа, ориентированная на дальнейшее обучение, обычно предоставляет четкий путь перехода к четырехлетнему обучению и достаточное количество курсов для подготовки студентов к более сложным углубленным курсам четырехлетней программы. В результате, студенты, пришедшие в четырехлетний институт из двухлетнего, способны переходить к университетской программе без потери курсов, наравне со своими сверстниками, с самого начала учившимися в университете. Мы полагаем, что институты, которые основывают свои начальные программы на моделях, представленных в главах 7 и 8, хорошо подготовлены для разработки согласованных программ, призванных обеспечить гладкий переход от колледжа к институту.

ГЛАВА 10. ПРОФЕССИОНАЛЬНАЯ ПРАКТИКА И ПРОФЕССИОНАЛИЗМ

Сегодня, когда мы вступаем в XXI век, существует беспрецедентная возможность сделать обучение профессиональному подходу к информатике равноправной задачей учебной программы. Понимание важности профессионализма критично для большинства студентов, так как основная часть выпускников реализовывает себя в индустрии. В этой главе обсуждаются различные пути введения соответствующей практики в курс информатики. Отдельные разделы рассматривают основные предпосылки, текущее состояние дел, поддержку профессионализма бизнесом и государственным сектором, методы внедрения профессиональной практики в учебную программу и стратегии оценки эффективности этих методов.

10.1. Предпосылки

Необходимость введения профессионализма в программу обучения основывается на потребностях реального мира, таких как растущий спрос на высококачественные продукты, увеличивающийся уровень ответственности разработчиков ПО и необходимость в постоянном повышении квалификации по окончании учебного заведения. В большинстве случаев, студенты поступают в институт без понимания этих вопросов, что создает сложности для преподавателей и для будущих работодателей. Чем больше профессиональной практики получают студенты, тем привлекательнее становится для них учеба и реальнее будущая работа. Соответственно, профессиональная практика в учебной программе может служить своеобразным катализатором пробуждения и поддержания интереса студентов к информатике.

И частный, и государственный сектор, безусловно, заинтересованы в обучении студентов профессионализму. Студенты, знакомые с реалиями профессиональной деятельности, понимают значение навыков эффективного общения с коллегами и клиентами, прилагают все усилия для того, чтобы делать свою работу качественно, стремятся к постоянному повышению своей квалификации и усовершенствованию своей фирмы. Каждый год Национальная ассоциация колледжей и работодателей проводит исследования для выяснения, какие качества работников наиболее важны с точки зрения работодателей [NACE2001]. В 2001 году список десяти основных факторов включал:

1. Навыки эффективного общения (как устного, так и письменного)
2. Честность
3. Навыки работы в коллективе
4. Умение налаживать межличностные отношения
5. Мотивированность и инициативность
6. Развитая профессиональная этика
7. Аналитические навыки
8. Гибкость и адаптируемость
9. Навыки работы с компьютером
10. Уверенность в себе

Результаты этого и подобных исследований подчеркивают важность взгляда на профессионализм как на центральный компонент программы обучения.

Всевозрастающая потребность в высококачественных продуктах также стимулирует усиление роли профессионализма в процессе обучения. Анархичные подходы к написанию программного обеспечения завоевали устойчивую репутацию основного источника проблем. Результатом этого служат все возрастающие требования клиентов к организации процесса производства ПО у поставщиков, причем без удовлетворения этих требований невозможно получить контракты. В частности, после потерь миллионов долларов, выплаченных за неработающее или незаконченное программное обеспечение, государственные органы США, в том числе Министерство обороны, начали требовать от всех своих подрядчиков соответствия как минимум третьему уровню зрелости модели CMM [Paulk95]. Для соответствия этим стандартам подрядчики должны наладить стабильный, надежный и устойчивый процесс разработки программного обеспечения. Неудовлетворенные клиенты, в особенности, не связанные с подрядчиками длительными отношениями, часто пытаются решить свои проблемы через суд, требуя возврата денег, завершения проекта и компенсации ущерба. Студенты должны понять значимость налаживания личных контактов с клиентами, согласования детальных требований к будущим программам и необходимых усилий по достижению наивысшего уровня качества.

IEEE и ACM способствуют развитию профессиональной ответственности различными способами:

- Разрабатывают и содействуют развитию кодексов этики [ACM2001, IEEE2001, SEEPP98], которым должны следовать члены перечисленных организаций. Эти кодексы выдвигают на первый план честность, откры-

тость, следование наивысшим стандартам качества, лидерство, поддержку общественных интересов и самообразование.

- Спонсируют авторитетные подгруппы, Общество социальных последствий технологии (Society on Social Implications of Technology, SSIT) и специальная группа по компьютерам и обществу (Special Interest Group on Computers and Society, SIGCAS), которые специально изучают вопросы этики и профессионализма.

- Разрабатывают и обновляют рекомендации по составлению учебных планов, например, этот отчет и предшествовавшие ему.

- Участвуют в разработке рекомендаций по аккредитации, обеспечивая включение профессионализма как предмета в учебные программы [ABET2000, CSAB2000].

- Поддерживают студенческие организации, поощряя студентов развивать понимание профессиональной деятельности и свой уровень зрелости как профессионалов.

- Путем публикации технических изданий, организации конференций и выпуска учебных материалов они создают возможности для постоянного профессионального роста специалистов.

И студентам, и обществу следует знать, что они могут и должны ожидать от профессионалов в области информатики. Например, студенты должны понимать важность профессионального поведения на работе. Они также должны осознавать, что профессиональное сообщество, основываясь на выработанных нормах этики и понимании важности практического опыта, предлагает студентам поддержку в их становлении как профессионалов. Опираясь на эту поддержку, дополняющую формальное обучение, студенты могут избежать ощущения изолированности, обычно испытываемого молодыми специалистами, а также получить хорошую, зрелую и этически выверенную профессиональную позицию.

10.2. Профессионализм в сегодняшнем образовании

На сегодняшний день существует много стратегий по введению профессионализма в программу обучения. Одной из наиболее общих характеристик этих стратегий является использование курсов, которые помогают студентам развить свои навыки общения, способность к решению проблем и технические умения. Набор этих качеств может быть приобретен как на курсах информатики, так и на курсах вне факультета информатики, таких как занятия по публичным выступлениям на факультете психологии или занятия по оформлению технической документации на филологическом факультете. Однако органы аккредитации обычно требуют, чтобы студенты не только *приобретали* эти навыки, но и *применяли* свои умения в дальнейших курсах.

Уровень запланированной профессиональной практики варьируется в зависимости от задач института, ресурсов факультетов и возможностей кафедр. Например, в 1999 году Лори Кинг (Laurie King) с факультет математики и информатики в Holy Cross College, провела среди членов ACM SIGCSE неформальный опрос, касающийся введения этики в программу обучения. Из 74 ответивших учебных заведений только 40 отводили этике в своих программах достаточное количество времени, удовлетворяющее критериям CSAB2000. Многие институты явно считают этот материал несущественным, хотя таких оказалось менее половины, что внушает некоторый умеренный оптимизм. Учитывая растущее внимание к проблемам профессионализма в нашей области, весьма вероятно, что эти институты вскоре включат в свои программы обучение профессиональной этике.

Следующий список иллюстрирует потенциальные механизмы введения в программу дополнительного материала, развивающего профессионализм:

- *Заключительные проектные курсы.* Эти курсы обычно длятся один-два семестра в течение последнего года обучения. В них студенты разрабатывают и осуществляют проект, работая в команде. Этот проект должен учитывать значимые для реального мира факторы, такие как стоимость, безопасность, производительность и удовлетворенность заказчиков и пользователей. Проект может разрабатываться исключительно в учебных целях или же иметь реальных заказчиков, в качестве которых могут выступать, к примеру, другие факультеты института. Хотя в таких курсах делается особый упор на работу над проектом и дальнейшую презентацию проекта студентами, здесь также может рассматриваться материал об интеллектуальной собственности, авторских правах, патентах, праве, этике и т.п.

- *Курсы профессионализма, этики и права.* Эти курсы обычно длятся один семестр и раскрывают перед студентами вопросы профессиональной практики, норм этического поведения и юриспруденции. Изучаемыми темами могут быть история информатики, влияние компьютеризации на общество, карьера в области информатики, юридическая и моральная ответственности.

- *Производственная практика/стажировка.* Эти программы обычно спонсируются университетом (что предпочтительно) или факультетами и позволяют студентам получить опыт реальной производственной работы до окончания учебного заведения. По крайней мере, один или два координатора должны наблюдать за программой, один на уровне института, а другой, возможно, с неполной занятостью, на факультете. Студенты обычно работают в течение лета и/или от одного до трех семестров, идущих не подряд. Обычно производственная практика проходит вне учебного заведения и таким образом прерывает процесс обучения на лето или семестр. В большинстве случаев студенты получают зарплату за эту работу, но иногда также получают и зачетные баллы.

- *Проектные курсы, ориентированные на коллективную работу.* Эти курсы акцентируют внимание на процессе разработки программного обеспечения и обычно включают групповой проект. Темы, рассматриваемые в этих курсах, включают управление проектами, экономику, анализ рисков, управление требованиями, проектирование, внедрение, сопровождение и списание ПО, обеспечение качества программ, этику и работу в коллективе. Они обычно охватывают много материала, но без глубокого его изучения.

Многие курсы вне факультетов информатики также могут помочь студентам в развитии профессиональных качеств. Такие курсы включают в себя (но не ограничиваются) философию этики, управление бизнесом, экономику, техническое взаимодействие и инженерный дизайн.

10.3. Поддержка практики профессионализма

Поддержка включения профессиональной практики в программу обучения может осуществляться многими сторонами. Приведенные ниже подразделы рассматривают роль в этом процессе частного и государственного секторов, отношение между академической подготовкой и рабочей средой, роли администраций университетов, факультетов и самих студентов в превращении профессионализма в одно из приоритетных направлений образовательного процесса.

10.3.1. Частный и государственный секторы

Большинство студентов, окончивших высшие учебные заведения, получают работу в частном или государственном секторах. Как основные потребители специалистов, имеющих высшее образование, индустрия и правительство играют важную роль в помощи по развитию практического направления обучения профессионализму в университетах. Студенты, задействованные в производственной практике, быстрее "профессионально взрослеют" и начинают серьезней относиться к учебе. Производственная практика также может помочь студентам трудоустраиваться после окончания университета, а работодателям – заранее находить подходящих им сотрудников. При поддержке частного и государственного секторов охват студентов профессиональной практикой значительно увеличивается – как в виде аудиторных, так и в виде выездных занятий.

Один из наиболее важных путей поддержки образовательного процесса частным и государственным секторами – это вовлечение сотрудников предприятий в обучение студентов. Сотрудники предприятий могут предоставить поддержку по многим направлениям:

- Они могут выступать в роли наставников учащихся, работающих над проектом.
- Они могут читать лекции о своих предприятиях, работе и производственных процессах.
- Они могут ассистировать преподавателям, ведущим курсы.
- Они могут предоставлять студентам учебные и исследовательские материалы своих предприятий, а так же проводить корпоративные курсы и тренинги для студенческой аудитории.
- Они могут быть членами консультативных комитетов, и участвовать в конструктивных обсуждениях проблем факультета и студентов.

С помощью любого из этих подходов, учреждения частного или государственного сектора могут установить важные связи с учебными заведениями, обеспечивающими их будущими сотрудниками.

Кроме многочисленных возможностей проведения занятий на территории учебного заведения, промышленность и государство также могут сделать значительный вклад в развитие профессионализма среди студентов, путем предоставления студентам возможности учиться и работать за пределами обычной академической обстановки. Учащиеся и преподаватели могут совершать выезды в местные предприятия и начинать устанавливать более тесные отношения с ними. Со временем, такое сотрудничество с предприятиями вырабатывает у студентов лучшее понимание своей будущей профессиональной роли. Кроме того, учащиеся могут сильней заинтересоваться изучаемыми ими предметами, и этот обновленный интерес будет способствовать повышению их рыночного потенциала. Учащиеся могут также завязывать доверительные отношения с предприятиями, формируя предпосылки к своему дальнейшему там трудоустройству. Для профессорско-преподавательского состава возможности консультирования устанавливают более высокий уровень доверия между факультетом и предприятием. Таким обра-

зом, работники предприятий, студенты и преподаватели узнают больше друг о друге, и охотнее содействуют друг другу в дальнейшем.

И, конечно, одна из наиболее важных форм поддержки институтов частным или государственным секторами – это финансовая поддержка учебных заведений и профессиональных организаций, например, пожертвования или гранты в виде оборудования, программных средств, скидок на продукты, денег, времени и т.п. Зачастую такие пожертвования критически важны для модернизации ресурсов, таких как лабораторное оборудование и программное обеспечение, а также для финансирования дополнительных стипендий и премий учащимся и преподавателям за достижения в их деятельности. Такие дополнительные стипендии могут использоваться для организации студенческих соревнований по программированию и проектированию или учебных конкурсов. Гранты могут значительно расширить границы исследований и проектов. На этом уровне частный и государственный сектора помогают обеспечить жизнестойкость и прогресс образовательной системы, а также развитие информатики.

Благодаря тесному сотрудничеству, пониманию интересов и ценностей друг друга, а также терпению, частный/государственный сектор и образовательные учреждения могут работать вместе над воспитанием профессионалов высокого класса. Такой союз необходим для подготовки студентов, имеющих высокие этические стандарты и ответственно относящихся к нуждам пользователей своих будущих продуктов.

10.3.2. Моделирование локальной и интернациональной рабочих сред

Большинство представителей индустрии желают видеть "готовых к работе" выпускников. Аналогично, большинство выпускников рассчитывают приступить к работе без дополнительного обучения. Учебный опыт отличается от производственного, и преподаватели должны стремиться облегчить процесс перехода из учебного заведения в мир реального бизнеса:

- моделируя для студентов реальную рабочую среду;
- обучая их работать в команде;
- обеспечивая опытом участия в большом проекте.

Внесение этих элементов в учебный план помогает моделировать как локальную, так и интернациональную рабочие среды.

Поскольку новые вычислительные средства появляются чрезвычайно быстро и в реальном мире одновременно сосуществуют технологии различных поколений, невозможно предугадать, в какой именно среде будут работать сегодняшние студенты после окончания учебного заведения. Таким образом, неразумно фокусировать изложение материала на определенном наборе средств. Широкий охват платформ и программных средств обеспечивает лучшую и более гибкую профессиональную подготовку учащихся, не замыкая их в узком кругу одного привычного окружения.

Для многих студентов обучение работе в команде не является естественным процессом, но он крайне необходим. Учащиеся должны привыкать работать как большими, так и маленькими командами и овладевать приемами планирования, бюджетирования, организаторскими и коммуникационными навыками. Коллективная работа должна поддерживаться богатым лекционным материалом. Материал занятий может включать вопросы планирования проектов, способы повышения эффективности общения, характеристики успешных коллективов, анализ причин основных проблем, возникающих в группах, и т.п. Итоговая оценка может базироваться на результате командной работы, индивидуальных достижениях членов команды и комбинации этих показателей. Поведение отдельных студентов также может быть фактором, учитываемым при оценке.

Опыт участия в проекте может заметно развить навыки решения проблем, если перед студентами ставятся задачи, не разрешимые простым путем. Такие проекты могут быть как контролируемыми аудиторными занятиями, так и содержать элементы непредсказуемости и импровизации, если работа проводится на территории заказчика. Целью проекта является развитие студенческих навыков за рамками простого умения решать в одиночку отдельные задачи.

10.3.3. Роль администрации, профессорско-преподавательского состава и студентов

На уровне наивысшего руководства администрация должна мотивировать как профессиональную деятельность преподавателей, так и их усилия по развитию факультетов. Такая деятельность может включать консультационную работу, активность в рамках профессиональных сообществ, летнюю практику, получение сертификаций и профессиональных лицензий, работу по получению аккредитации, формирование индустриальных консультативных советов с соответствующими подразделениями, организация программ практики для студентов, установление связей с частным и государственным секторами. Эта деятельность особо трудоемка, но чрезвычайно важна и для индивидуумов, и для учреждений, которые должны принимать ее во внимание и всячески поощрять и стимулировать.

Преподаватели и студенты могут работать вместе, сообща поддерживая и развивая профессиональные этические и поведенческие стандарты. Преподаватели должны присоединяться к профессиональным сообществам и помогать учащимся в создании в их учебном заведении студенческих секций и отделений таких сообществ. Эти студенческие секции могут вручать награды за значительные достижения в учебе, общественную работу и активную практическую деятельность. В дополнение, студенческие секции могут организовывать встречи с потенциальными работодателями и способствовать получению институтами дотаций и других видов помощи со стороны.

10.4. Включение профессионализма в учебную программу

Включение профессионализма в учебную программу должно быть осознанным и продуманным шагом, поскольку в уже существующие курсы необходимо внедрять большое количество материала. Например, вводные курсы могут содержать обсуждения и задания на тему влияния информатики на общество и важности профессионализма в практической деятельности. Достигнув уровня второго курса, студенты могут начинать делать отчеты о проделанной работе, как это должны делать профессионалы, в виде требований, проектных и тестовых документов.

Дополнительные материалы, такие как история информатики, подходы к решению нечетко сформулированных задач, командная работа с индивидуальной ответственностью, этические проблемы реальной жизни, стандарты и рекомендации, философские основы этических суждений и т.п., также могут быть рассмотрены в отдельных курсах или же распределены по множеству основных курсов. Последний подход имеет преимущество в представлении материала в контексте реальной области применения. С другой стороны, он несколько проблематичен, поскольку охват вопросов профессиональной практической деятельности сокращается в борьбе за достаточное количество времени, необходимого также и для изложения технического материала. Проектные курсы обеспечивают естественный каркас для большей части необходимого практического материала, особенно если факультеты могут привлекать внешних заказчиков, нуждающихся в не критически важных системах. При занятости в проектах, ориентированных на предоставление услуг или работу с клиентами, учащиеся начинают понимать необходимость этичного поведения в различных условиях. В результате учащиеся узнают больше о том, как лучше подойти к потребностям клиента и решению его задачи. Не имеет значения, насколько практика профессионализма интегрирована в основной курс обучения, но крайне важно, чтобы соответствующий материал был закреплен упражнениями, проектами и экзаменами.

Факультеты с адекватным преподавательским составом и ресурсами, могут уделить больше внимания курсам, посвященным преподаванию профессиональной практической деятельности. Можно читать отдельные курсы по вопросам практики, этике, законодательству, а так же организовать дополнительные проектные курсы. Можно так же предлагать студентам более глубокие курсы по экономике информатики, качеству, надежности, безопасности и т.п. Как указывалось ранее в разделе 10.2, эти курсы могут быть из дисциплин вне информатики и, тем не менее, они будут оказывать существенное влияние на профессиональное становление студентов.

10.5. Оценка профессионализма

Преподаватели могут обеспечить результативность профессиональной практики путем создания инфраструктуры, в которой результаты учащегося оцениваются согласно индустриальным стандартам и активно поощряется профессионально выполненная работа. Такая инфраструктура должна строиться на следующих аспектах.

- Оценка, основанная на результатах.
- Пересмотр заданий, проектов и экзаменов с точки зрения включения достаточного объема материала по профессионализму.
- Конструктивное обсуждение и объективное измерение студенческой работы с целью демонстрации и оценки прогресса.
- Привлечение учащихся к обсуждению и оцениванию выполненных работ для выработки лучшего понимания сути таких обсуждений и оценок.
- Привлечение профессионалов частного или государственного секторов к оценке проектной работы студентов.
- Использование стандартизированных тестов для отслеживания общего прогресса студентов.
- Проведение среди выпускников опросов с целью выяснения, насколько их образование помогло в карьере.
- Выполнение аккредитационных требований для подтверждения должного уровня.

Процесс оценки должен стимулировать студентов к хорошей технической практике и высшим стандартам подготовки. Например, должно осуждаться желание завершить работу с плохой подготовкой и малым количеством времени (выполнение работы в ночь перед экзаменом). В процессе оценки учащиеся должны полагаться на свои индивидуальные знания, даже если работали командой. Процесс оценивания должен базироваться на объективных метриках, к использованию которых студенты привыкают и начинают использовать их самостоятельно для отслеживания своего прогресса.

ГЛАВА 11. ХАРАКТЕРИСТИКИ ВЫПУСКНИКОВ ФАКУЛЬТЕТОВ ИНФОРМАТИКИ

Первый критерий проверки университетской программы обучения в области информатики – это соответствие совокупности знаний по информатике. Однако существуют и другие вопросы, требующие рассмотрения. Эти вопросы касаются, например, общей сущности дисциплины, объема и глубины программы, а также других факторов, относящихся к индивидуальным и практическим навыкам и знаниям.

Вообще говоря, учебные заведения должны формально описывать цели и ожидаемые результаты своих учебных программ. Существует множество различных образовательных программ в области информатики, каждая из которых готовит учащихся к разным, но одинаково полноценным профессиям. Одним полюсом являются образовательные программы, которые обеспечивают учащимся возможность изучить широкий спектр разделов информатики, что впоследствии поможет им легче находить пути решения задач в различных областях. Кроме того, именно такие специалисты первыми начинают работать в новых, только появившихся областях дисциплины. Другой полюс – программы, рассматривающие одну выбранную область информатики и глубоко раскрывающие ее суть, что позволяет выпускникам иметь набор прочных знаний в области выбранной ими специализации, будь то разработка мультимедийных систем, проектирование сетей или любая другая специальность. Однако, в любом случае, несмотря на разницу в акцентах и содержании различных программ, существует минимальный набор определенных качеств выпускников ВУЗов, получивших образование в области информатики. В данной главе мы попробуем исследовать эти качества.

Материал этой главы во многом основан на отчете, подготовленном в Великобритании с целью определения желаемых качеств выпускников в области информатики [QAA2000]. Его задача заключалась в определении порогового уровня требований ко всем специалистам в данной области. Конечно, описание характеристик выпускников, помещенное в этой главе, основывается на том же фундаменте, что и цели образовательных программ, описанные в предшествующих главах. Разница, главным образом, в точке зрения и перспективе. Анализ целей обучения в институтах в терминах требований к выпускникам упрощает разработку системы оценок, проверяющих достижение поставленных целей.

11.1. Общие характеристики выпускников факультетов информатики

Хотя характеристики выпускников факультетов информатики связаны с целями обучения обязательных разделов, наши ожидания от выпускников факультетов информатики предполагают значительно больший уровень достижений. Цели обучения, детально рассмотренные в Приложении А, описывают, что именно должны знать студенты после завершения того или иного модуля. Задача этого раздела – определить конкретные качества, которыми должны обладать успешные выпускники. Эти качества включают в себя:

- *Системный взгляд на дисциплину.* Цели обучения, связанные с конкретными модулями знаний, имеют тенденцию фокусироваться на отдельных концепциях и темах, что впоследствии может привести к фрагментарному усвоению дисциплины. Обучающиеся должны развить в себе высокоуровневое понимание систем в целом. Это восприятие должно преодолевать детали отдельных реализаций различных компонент и давать общее понимание структуры компьютерных систем и процессов их создания и анализа.

- *Понимание связи теории и практики.* Фундаментальный аспект информатики – это равновесие между теорией и практикой, их тесная связь друг с другом. Выпускники должны четко понимать не только теоретическую часть материала, но и влияние теории на практику.

- *Твердое владение основными методами информатики.* В процессе обучения студенты сталкиваются со многими общими методами, такими как абстракция, рекурсия и эволюционные изменения. Выпускники должны осознавать широту применения этих методов в области информатики и не сводить их применимость только к тому материалу, в рамках которого они были представлены.

- *Опыт участия в большом проекте.* Для того чтобы выпускники умели грамотно применять полученные знания, они обязательно должны принять участие хотя бы в одном реальном проекте. Такого рода опыт обучает студентов практически использовать приобретенные навыки и заставляет студентов интегрировать материал, изученный на различных курсах.

- *Адаптируемость.* Одной из основных характеристик информатики на протяжении всей ее относительно небольшой истории является очень быстрый темп изменений. Поэтому выпускники должны обладать глубокими фундаментальными знаниями, помогающими им вырабатывать новые необходимые навыки по мере того, как эволюционирует область. Стратегии достижения успеха в этом направлении описаны в разделе 11.3.

11.2. Профессиональные качества и способности

Изучающие информатику студенты должны развить широкий диапазон профессиональных качеств. Некоторые из этих качеств специфичны для информатики, в то время как другие носят более общий характер и считаются обязательными для выпускников любых технических специальностей. Обсуждаемые качества могут быть разделены на три основные категории:

- Когнитивные качества, относящиеся к специфическим для информатики видам интеллектуальной деятельности
- Практические знания, связанные с информатикой
- Дополнительные качества, возможно, развитые в контексте информатики, но имеющие общий характер и применимые также и в других контекстах

Необходимые качества и способности описаны в Таблице 11-1. Любое высшее учебное заведение должно обеспечить достаточное внимание развитию у студентов качеств каждой категории – когнитивных, практических и общих – до выпуска из университета.

Таблица 11-1. Профессиональные качества выпускников

Когнитивные навыки, связанные с информатикой.

- *Знания и понимание*. Демонстрация знаний и понимания основных фактов, концепций, принципов и теорий, связанных с информатикой.
- *Моделирование*. Использование полученных навыков в моделировании и проектировании информационных систем с демонстрацией способности выбора правильных компромиссных решений.
- *Требования.* Выявление и анализ критериев и требований, относящихся к конкретным задачам, планирование стратегий их решения.
- *Критическая оценка и тестирование.* Анализ того, насколько конкретная информационная система отвечает критериям, определенным для ее использования и будущего развития.
- *Методы и средства.* Использование соответствующих теоретических знаний, практических навыков и инструментов для проектирования, реализации и оценки компьютерных систем.
- *Профессиональная ответственность.* Следование профессиональным, социальным и этическим нормам, касающихся области компьютерных технологий.

Практические навыки, связанные с информатикой

- *Проектирование и реализация.* Спецификация, проектирование и реализация компьютерных систем.
- *Оценка*. Оценка систем и их качественных характеристик, возможных компромиссных путей решения конкретной задачи.
- *Управление информацией*. Применение принципов эффективного управления информацией к различным видам информации, включая текстовую, графическую, видео и звуковую.
- *Человеко-машинное взаимодействие.* Применение принципов человеко-машинного взаимодействия при оценке и создании широкого диапазона продуктов, включая пользовательские интерфейсы, web-страницы и мультимедийные системы.
- *Оценка риска.* Определение рисков и связанных с вопросами безопасности аспектов эксплуатации компьютерного оборудования в заданном контексте.
- *Инструменты и средства.* Эффективное использование адекватных инструментов при разработке и документировании ПО, с акцентом на полном понимании процесса решения практических задач с помощью компьютера.
- *Эксплуатация.* Эффективная эксплуатация компьютерного оборудования и программных средств.

Дополнительные качества:

- *Общение.* Способность публично выступать перед различными аудиториями с докладами/сообщениями о технических проблемах и путях их решения
- *Командная работа.* Умение эффективно работать в производственном окружении
- *Способность к количественному мышлению.* Понимание и объяснение количественных характеристик проблемы.
- *Самоуправление*. Управление собственным обучением и развитием, управление временем и организаторские качества.
- *Профессиональное развитие*. Стремление всегда быть в курсе текущего состояния дел в дисциплине, продолжать свое профессиональное развитие.

11.3. Готовность к изменениям

Любая учебная программа по информатике должна научить выпускников справляться с трудностями, вызванными быстрым темпом изменений в компьютерной сфере, и даже извлекать из этого пользу. Но как достичь в этом успеха на практике? С одной стороны, скорость изменений требует постоянного обновления учебной программы и оборудования. С другой стороны, она мотивирует к изменению стратегии преподавания, смещению фокуса с конкретных быстроустаревающих технологий, к моделям обучения, которые поощряют студентов самостоятельно приобретать новые знания и навыки.

Для того чтобы научить студентов справляться с изменениями, необходимо привить им такое отношение к учебе, которое обеспечит их стремление к самосовершенствованию на протяжении всей карьеры. Поэтому программа обучения информатике должна выполнять следующие требования:

- Применение методики преподавания, которая подчеркивает различие между преподаванием и (само)обучением, стимулирует студентов мыслить независимо
- Обучение студентов на творческих задачах и упражнениях, развивающих их инициативность.
- Использование методически согласованных теоретических и практических курсов, что обеспечивает стабильность закрепления материала
- Постоянное обновление оборудования и программного обеспечения
- Ознакомление студентов с информационными ресурсами и стратегиями обновления своих знаний
- Поощрение коллективного обучения и использование телекоммуникационных технологий для обеспечения взаимодействий групп учащихся

- Убеждение студентов в необходимости продолжения профессионального развития и самосовершенствования на протяжении всей жизни

11.4. Стандарты аттестации

В поисках набора критериев, подходящего для оценки уровня выпускников факультетов информатики, авторы доклада об аттестации [QAA2000] обнаружили, что установление минимальных требований к учащимся может стать причиной того, что студенты и преподаватели перестанут стремиться к уровню совершенства, превышающему установленный минимум. Во избежание этого предлагается оценивать различные уровни достижений. Для самого низшего уровня рассматривается "пороговый стандарт", состоящий из минимального набора требований, которым должен удовлетворять любой выпускник. В упомянутом отчете также рассматривается "модальный стандарт", которому должны отвечать знания среднего студента.

Определение факультетом требований к каждому из этих двух стандартов предоставляет ценную возможность оценить общее значение излагаемого материала и эффективность образовательного процесса. Хотя эти стандарты будут варьироваться в зависимости от конкретной программы и специфики института, характеристики, позаимствованные из упомянутого выше доклада (см. Таблицу 11-2) могут предоставить полезную модель для создания локальных стандартов.

Таблица 11-2

Минимальный (пороговый) уровень

- Демонстрация общего понимания основной области знаний и теории информатики
- Понимание и применение основных концепций, принципов и практических приемов в контексте конкретной задачи, демонстрация способности к адекватному выбору используемых методик
- Выполнение работы, включающей идентификацию проблемы, ее анализ, проектирование и реализацию соответствующей программной системы, а также подготовку необходимой документации. Работа должна демонстрировать навыки решения задач и понимание важности качества итогового продукта.
- Демонстрация умения работать как индивидуально под руководством, так и в команде
- Понимание сложившихся профессиональных, юридических и этических практик
- Понимание необходимости постоянного саморазвития
- Понимание области применения полученных знаний

Средний (модальный) уровень

- Демонстрация глубокого понимания основных разделов всей области знаний информатики, равно как и способности к сравнительному/критическому анализу различных подходов
- Выбор и применение диапазона концепций, принципов и практических приемов в контексте нечетко поставленной задачи, демонстрирующий эффективность принятия решений в применении инструментов и методик
- Выполнение работы, включающей описание проблемы, ее анализ, проектирование и разработку соответствующей системы программного обеспечения, а также подготовку необходимой документации. Работа должна показать широкий диапазон навыков решения задач и высокое качество
- Демонстрация умения работать как индивидуально, так и в качестве лидера либо члена команды
- Следование принятым практикам в профессиональной, юридической и этической сферах
- Знание механизмов постоянного самосовершенствования в профессиональной сфере
- Понимание области применения полученных знаний

Хотя эти стандарты определены только для минимального и среднего уровней, важно, чтобы институты предоставляли талантливым студентам возможность полного раскрытия своего потенциала. У таких студентов нужно развивать творческий и новаторский подход к применению знаний, умение создавать сложные программные системы и достигать поставленные перед ними цели, делать научную работу, конструктивно анализировать и обсуждать как собственные достижения, так и результаты коллег. Человеческие изобретательность и творчество послужили причиной быстрых темпов развития информатики в последние годы, и сегодняшние учебные программы не должны ограничивать тех, кто будет творить эту дисциплину завтра.

ГЛАВА 12. ИНФОРМАТИКА В УЧЕБНЫХ ПЛАНАХ

Как описано в главе 1, специальная комиссия по подготовке отчета CC2001 назначила ряд рабочих групп по педагогике для рассмотрения учебных планов с более общей точки зрения, чем это делали группы по структуризации знаний. Заботясь о целостности подхода, рабочие группы по педагогике должны были определить те общие мотивы и тенденции, которые могли остаться незамеченными при тематическом рассмотрении дисциплины.

Большинству рабочих групп было поручено составление рекомендаций, относящихся к специфическим аспектам учебного плана информатики. Рабочая группа по *информатике в учебных планах* имела более содержательное задание, частично состоявшее в изучении аспектов информатики, имеющих отношение ко всем гражданам и академическим дисциплинам, и в предложении рекомендаций по использованию информатики в обучении студентов различных специальностей.

Данная глава представляет собой отчет этой рабочей группы, посвященный обязанностям факультетов информатики перед своими отдельными коллегами и академическим сообществом в целом. В этом докладе рабочая группа интерпретирует фразу "информатика в учебных планах" как описание учебных планов (т.е. курсов или курсовых модулей), нацеленных на студентов других дисциплин. Хотя студенты, изучающие информатику, могут записываться на такие курсы, изначально они создаются для учащихся других специальностей. Эти курсы являются одним из способов, которыми информатика как дисциплина исследует и раскрывает свою природу.

Данная глава разделена на три части. В разделе 12.1 мы обсуждаем важную роль общеобразовательных курсов и обосновываем посылку, что разработка и преподавание таких курсов должны рассматриваться как часть миссии факультетов информатики. В разделе 12.2 мы в общих чертах описываем процесс спецификации, разработки, реализации и оценки курсов. Этот раздел освещает вопросы, посвященные созданию новых курсов. Соответствующий список вопросов далеко не полон, но может быть использован как отправная точка, определяющая значимые образовательные цели. В разделе 12.3 мы определяем и описываем три различных формата курсов, которые могут предлагать факультеты информатики.

12.1. Мотивация и цели

Факультеты информатики обычно существуют как части учебных институтов, имеющих в своем составе также естественнонаучные и гуманитарные факультеты, факультеты социальных наук и прикладного искусства. В предыдущих главах уже было уделено немало внимания стремительному развитию информатики и последствиям этого развития, отразившимся практически в каждой области человеческого знания. На сегодняшний день информатика является не просто важной дисциплиной, она также обслуживает множество различных других наук. Муниципальный чиновник, работающий с демографической базой данных, дизайнер, использующий автоматизированную систему проектирования, экономист, создающий компьютерные модели – это все примеры людей, применяющих информационные технологии в своей профессиональной деятельности. Столь широкое распространение информатики влечет за собой возможность и необходимость создания множества высококачественных учебных курсов для заинтересованных специалистов из иных областей. Помня о своей главной задаче – подготовке профессионалов в информатике – мы не должны забывать о других.

Вследствие ограниченности финансовых ресурсов и нехватки преподавателей факультеты информатики иногда вынуждены концентрировать собственные ограниченные возможности на своих студентах в ущерб учащимся других специальностей. Мы считаем, что такая политика неуместна и недостойна. Учитывая скорость распространения влияния информатики на все сферы жизни, каждый университет должен предоставить курсы обучения этой дисциплине всем своим студентам. Так как наиболее эффективно такие курсы могут преподавать именно факультеты информатики, то они должен иметь достаточно ресурсов для: 1) обучения своих студентов, 2) помощи студентам других факультетов в понимании и применении информатики. Обе эти задачи крайне важны.

12.2. Процесс создания курсов

Полезная модель для разработки курсов – процесс разработки программного обеспечения (ПО). Так же, как и в случае с ПО, процесс разработки курсов может быть поделен на четыре стадии: спецификация, проектирование, применение и оценка. Каждая из этих стадий рассмотрена в следующих подразделах.

12.2.1. Цели курсов (спецификация курса)

При разработке общеобразовательных курсов необходимость задать множество важных вопросов и затем ответить на них. Но кому мы должны адресовать эти вопросы? Кто должен формулировать цели и описывать содержание общеобразовательных курсов по информатике? Конечно, факультеты информатики должны помогать специфицировать цели таких курсов, но именно помогать, а не диктовать. Очень важно добиться того, чтобы обсуждение этих курсов велось как специалистами-информатиками, так и профессионалами из других областей, чтобы цели курсов отражали реальные потребности учащихся, а не желание преподавателей обучать тем или иным

предметам. В прошлом, факультеты математики часто подвергались критике за создание таких вводных курсов, которые концентрировались в основном на абстрактных темах "чистой" математики, хотя большинству студентов были интересны и важны прикладные темы. Информатика не должна повторить эту ошибку. Предлагая свою помощь в формулировке целей курсов, мы должны внимательно прислушиваться к потребностям студентов и преподавателей других факультетов.

Существует четыре основные цели общеобразовательных курсов:

1. Удовлетворить интерес студентов, желающих ближе познакомиться с компьютерами.
2. Удовлетворить требования институтов к образованию студентов физических и/или математических специальностей.
3. Дать студентам знания и опыт использования информатики в своих дисциплинах.
4. Обеспечить понимание студентами информационных технологий, необходимых для активного участия в жизни современного общества.

Такие курсы могут преподаваться для широкой аудитории студентов, или же для группы учащихся, специализирующихся в одной области. Например, большинство институтов выдвигает некоторые требования к уровню компьютерной грамотности всех студентов. С другой стороны, факультет информатики может принять решение читать курсы по компьютерной графике только студентам, изучающим искусство. Очевидно, что когда курс разрабатывается для узкой группы студентов, соответствующий факультет должен достаточно подробно сформулировать свои требования к программе занятий. Мы убеждены, что это верно и для курсов с более широкой целевой аудиторией.

Первым шагом в разработке новых компьютерных курсов является определение требований учебного плана, которые на текущий момент не удовлетворены. Это можно делать либо постфактум, либо упредительно. Факультеты информатики определенно должны реагировать на потребности других факультетов либо промышленности в новых курсах, которые могут быть полезны с точки зрения студентов или работодателей. С другой стороны, факультеты информатики могут сами предлагать другим факультетам свои услуги по разработке курсов, включающих новый материал. Независимо от того, как был инициирован процесс разработки нового курса, если все вовлеченные стороны проявили интерес, то следующим шагом является определение целевой аудитории и выяснение пожеланий по поводу структуры и содержания курса. В этот момент должны быть изучены следующие вопросы:

- Каким потребностям, которые не были удовлетворены существующими курсами, будет отвечать разрабатываемый курс?

- На кого рассчитан новый курс? Какие факультеты и специальности заинтересованы в нем? Какие категории студентов будут его посещать? Есть ли способ количественно оценить интерес к курсу и потребность в нем? Будут ли студенты, на которых мы ориентируемся, иметь время и возможность посещать этот курс без ущерба для своих обязательных предметов?

- Как преподавание нового курса повлияет на наш факультет? Будет ли это сказываться негативно на наших основных курсах?

- Какое значение для студентов будет иметь факт прослушивания курсов? Заработают ли студенты зачетные баллы, а если да, то сколько и какого типа? Будет ли курс обязательным для студентов, факультативным или, может быть, ориентированным только на дальнейшее образование?

- Кто будет преподавать новый курс? Это будет один человек или группа сменяющих друг друга преподавателей? Достаточно ли силен и многочислен наш кадровый состав для организации такой группы, в том числе, с учетом планируемых отпусков и возможных внезапных болезней? Если нужного количества преподавателей нет, какие меры мы можем предпринять?

В дополнение к обработке собранных ответов на эти и подобные вопросы, факультет должен внимательно изучить отчет Национального Исследовательского Совета "О влиянии информационных технологий" [CSTB99]. Этот документ описывает фундаментальные цели и задачи общеобразовательных курсов по информатике и превосходно показывает связи информатики и учебного плана, а также предоставляет необходимые данные для разработки курсов.

12.2.2. Разработка курсов

Как только потребности учебного плана четко определены, и все факультеты поддержали инициативу создания курса, удовлетворяющего эти потребности, следующим шагом является разработка курса. Он состоит из уточнения основных целей и задач курса путем указания четких технических знаний и концепций, которые должны быть включены в программу, а также результатов, которые должны получить слушатели курсов. В этот момент важно ответить на следующие вопросы:

- Какие именно навыки должны получить слушатели курсов, в какой степени они важны для конкретной области? Какой глубины знаний мы хотим добиться от учащихся?
- Какие фундаментальные концепции информатики должны быть включены в курс и как эти концепции можно связать с уже преподававшимися?
- Должны ли присутствовать в программе курса в дополнение к техническому материалу социальные и этические вопросы? Если да, то какие именно?

Ранее мы ссылались на отчет Национального Исследовательского Совета "О влиянии информационных технологий" [CSTB99]. В нем определены три основные категории знаний, которые могут быть кандидатами на включение в программу общеобразовательного курса:

- *Навыки работы с компьютером.* Этот класс знаний включает в себя умение использовать распространенные программы – текстовые процессоры, Internet-браузеры, программирование в MathLab и т.д. Эти навыки должны быть четко описаны при проектировании курса. Но такого рода знания очень быстро устаревают, и учебный план курса будет нуждаться в периодическом обновлении.

- *Фундаментальные концепции информатики.* Как отмечено в вышеупомянутом отчете "концепции дают ответы на все вопросы "как" и "почему" информационных технологий и помогают осознать существующие возможности и ограничения. Концепции – ценный материал для понимания новых технологий по мере их эволюционирования". Основные концепции содержат общие идеи, которые инвариантны по отношению к производителям ПО, конкретным программным пакетам и узкоспециальным умениям. Примеры могут включать теорию алгоритмов, оценку сложности, архитектуру ЭВМ, способы представления информации, моделирование и т.п. Понимание фундаментальных основ информатики является исключительно важным для эффективной работы с компьютером. Важность специфических навыков мимолетна, в то время как знания фундаментальные концепций будут помогать студентам в течение многих лет, что особенно важно с учетом современных темпов изменений информационных технологий.

- *Общие интеллектуальные качества.* Этот класс знаний состоит из общих интеллектуальных навыков, важных для любой области обучения. Соответствующие навыки помогают студентам эффективно применять информационные технологии в решении сложных задач. Примеры включают отладку, устранение неполадок, логическое обоснование, навыки устной и письменной коммуникации и т.п. Эти качества важны и полезны для всех студентов, помогая им развивать и улучшать их интеллектуальные способности.

Отчет "О влиянии информационных технологий" приводит примеры всех трех категорий знаний и подчеркивает, что хорошо разработанный курс должен включать в себя все три категории, иначе курс быстро устаревает либо не обеспечивает должный уровень обучения требуемым знаниям и умениям. Баланс между всеми тремя направлениями крайне важен.

12.2.3. Внедрение курсов

После определения целей и содержания разработчики курса должны поставить перед собой вопросы, касающиеся его практической реализации:

- Должен ли курс предлагаться в формате общих лекций или узких дискуссий, семинаров? Нужны ли лабораторные работы? Насколько формальными они должны быть?

- Какие методики обучения наиболее соответствуют материалу курса? Нужна ли студентам в рамках курса работа над проектами? Сколько проектов потребуется и сколь объемными они должны быть? Командная работа? По завершении проектов – письменные отчеты или устные выступления?

- Как наиболее эффективно мы можем оценить результаты студентов? Какие типы проектных работ и/или экзаменов имеют для курса наибольший смысл?

- Какой уровень квалификации требуется для чтения курса? Есть ли у нас преподаватели, способные провести курс в одиночку, или потребуются усилия целой группы?

- Имеем ли мы адекватные ресурсы (оборудование, лаборатории и т.д.) для открытия таких курсов?

- Насколько студенты будут заинтересованы в курсах? Сколько раз в неделю должны проводиться занятия?

Ответы на эти и другие практические вопросы часто будут определяться не высшими академическими ценностями, но местными особенностями и ресурсными ограничениями, лежащими вне сферы компетенции данного документа. Эти факторы могут включать в себя финансовые вопросы, количество обучающихся студентов, требования института, интересы факультетов, ограничение помещений и даже соображения политического характера.

Но, независимо от сложности возникающих на этом пути проблем, факультет информатики должен обеспечить студентов общеобразовательными курсами соответствующими стоящим перед ними целям.

12.2.4. Оценка курсов

После завершения разработки курса, факультет готов предложить студентам новый общеобразовательный курс. Остается только финальный шаг – оценка курса. После того, как курс был прочитан два или три раза, он может быть осторожно пересмотрен и, возможно, изменен. Информация, необходимая для оценки курсов, собирается различными методами: письменные отзывы студентов, аудиторные наблюдения, индивидуальные интервью со студентами и преподавателями факультетов-заказчиков курса. Если курс читается уже несколько лет, то имеет смысл также проинтервьюировать выпускников, выясняя их мнение о пользе материала курса в повседневной работе.

Следующие вопросы касаются оценки курса:

- Отвечает ли курс стоящим перед ним целям? Если нет, нужно ли его переделать или заменить альтернативным?
- Была ли упущена какая-либо важная тема? Уделялось ли внимание чему-то ненужному?
- Базируясь на результатах тестирования, можно ли утверждать, что учащиеся достигли желаемого уровня знаний?
- Удовлетворен ли факультет-заказчик? Если нет, то что нужно предпринять для изменения ситуации?

Разработка и внедрение курсов не является одношаговым процессом, скорее это постоянный процесс, который, подобно процессу разработки ПО, может включать постоянные пересмотры и обновления. Такой подход особенно важен в связи с быстрыми темпами изменений в нашей области.

12.3. Модели курсов

Мы выявили три основных типа курсов, которые могут предлагаться факультетами информатики: общеобразовательные, мультидисциплинарные и узкоспециальные. Эти три подхода описаны в нижеследующих подразделах.

12.3.1. Общеобразовательные курсы

Такие курсы рассчитаны на всех учащихся и разработаны с целью удовлетворения интересов и потребностей всех желающих в познании информатики, а не с целью дать специфические знания в области конкретной дисциплины. Общеобразовательные курсы не заботятся о студенческих навыках работы с компьютером, вместо этого они предоставляют общую картину информатики, позволяя поднять уровень эрудированности студентов и воспитать более информированных граждан.

Один из вариантов таких курсов – это широкое описание дисциплины информатики, похожее на вводный курс $CS100_B$, описанный в Приложении Б. Другой возможностью было бы широкое введение в сетевые технологии и телекоммуникации, включающее как концептуальные, так и технические вопросы, а также обсуждение воздействия телекоммуникационных технологий на общество. Еще одним примером общеобразовательных курсов может служить курс под названием "Информатика и этика", целью которого является рассмотрение влияния информатики на общество, а также этические, социальные и моральные нормы в информатике – безусловно, важные и интересные темы для основного большинства студентов.

12.3.2. Мультидисциплинарные курсы

Такие курсы служат нескольким факультетам со схожими потребностями студентов в компьютерных знаниях. Общей характеристикой таких курсов является необходимость в некоторой специфической предварительной подготовке, не связанной напрямую с информатикой. Примеры мультидисциплинарных курсов:

- Курсы по численным методам для студентов естественнонаучных факультетов
- Курсы информационного моделирования для финансистов, экономистов, менеджеров и бизнесменов
- Курсы искусственного интеллекта для психологов, лингвистов и философов
- Курсы компьютерной графики для студентов, изучающих искусство

Отчет "О влиянии информационных технологий" также приводит примеры мультидисциплинарных курсов. Одним из них является курс применения информационных технологий в социальных исследованиях, включающий

изучение СУБД, методов поиска информации в Интернете, методов анализа данных и статистического ПО. Такой курс очевидно полезен студентам многих гуманитарных специальностей, например социологам, антропологам и политологам.

Факультеты информатики могут предлагать другим факультетам совместную разработку таких курсов, или же сами заинтересованные факультеты могут инициировать подобную деятельность. Вести мультидисциплинарные курсы могут как преподаватели факультетов информатики, так и смешанные команды представителей нескольких факультетов.

В небольших учебных заведениях, таких как частные гуманитарные колледжи, междисциплинарные курсы имеют больше шансов на успех, если они имеют общеобразовательный характер и привлекательны сразу для многих факультетов. Например, курс по вычислительной физике может не иметь смысла для небольшого института, в то время как курс по вычислительным наукам привлечет не только физиков, но и биологов, химиков, геологов и т.п.

12.3.3. Узкоспециальные курсы

Эти курсы более узко концентрируются на определенной дисциплине, чем рассмотренные выше, и обычно рассчитаны на однородную группу студентов одного факультета или даже одной специальности. К примеру, многие из нас знакомы с курсом дискретной математики, предлагаемым математическим факультетом специально для студентов факультета информатики. Такой тип курсов попадает в категорию узкоспециальных. Еще один пример – упомянутый выше курс вычислительной физики, или курс вычислительной биологии. Отчет "О влиянии информационных технологий" описывает пример курса для экономистов, в процессе которого студенты разрабатывают компьютерные модели экономических проблем или исторических событий, демонстрирующие влияние различных факторов на конечный результат. Курсы такого рода могут разрабатываться и читаться вместе факультетом информатики и профилирующим факультетом, что обеспечит должное внимание ко всем аспектам материала. В любом случае, нельзя забывать, что это общеобразовательные курсы, и в добавление к специфическим техническим знаниям необходимо обучать студентов фундаментальным концепциям информатики.

12.4. Заключение

В этой главе мы обосновали необходимость общеобразовательных курсов, предоставили рекомендации к разработке, внедрению и оцениванию курсов, описали три основные заслуживающие внимания типа курсов. При разработке курсов необходимо чутко прислушиваться к потребностям целевой аудитории и стараться максимально их удовлетворить, для чего нужно активно взаимодействовать с коллегами с других факультетов.

Большинство студентов из других дисциплин прослушивает только один курс по информатике. Поэтому очень важно, чтобы этот единственный курс был тщательно продуманным и максимально полезным для студентов. В нем должны быть широко представлены как практические навыки работы с компьютером, так и фундаментальные концепции информатики, что позволит студентам получить полное и прочное понимание материала.

ГЛАВА 13. ИНСТИТУЦИОННЫЕ ПРОБЛЕМЫ

Данный отчет разработан главным образом для колледжей и университетов, желающих совершенствовать и улучшать свою программу по информатике. С этой точки зрения, приложения к настоящему отчету предлагают глубокий анализ структуры и границ предмета информатика с детальным набором описаний курсов, которые предоставляют гибкие подходы к составлению учебного плана по информатике. Для составления успешной программы обучения каждый институт должен рассмотреть широкий спектр стратегических и тактических вопросов. Целью этой главы является выявление этих вопросов и рассмотрение их влияния на разработку учебного плана.

13.1. Локальная адаптация

Задачи разработки программы обучения различны в разных учебных заведениях из-за индивидуальных характеристик каждого института. И хотя большинство институтов соглашаются с общим набором требований к знаниям и умениям студентов, существует множество дополнительных факторов, влияющих на разработку программы. Рассмотрим эти факторы:

- *Типы институтов и ожидания от программы.* Как обсуждалось в разделе 9.3, институты различны по своей структуре и по требованиям к учебной программе. Количество обязательных курсов для студентов, специализирующихся в информатике, может отличаться почти в два раза в зависимости от типа института. Программа, идеально подходящая для гуманитарных колледжей США, может быть абсолютно неприемлемой для исследовательских университетов любой страны.

- *Диапазон последипломных целей, преследуемых выпускниками.* Институты, главной задачей которых является подготовка квалифицированной рабочей силы для предприятий, по-видимому, имеют иные цели при разработке программы обучения, по сравнению с институтами, готовящими студентов-исследователей к продолжению обучения. Учебные заведения должны осознавать, что их программа должна давать студентам подготовку, достаточную как для последующей научной карьеры, так и для профессиональной работы.

- *Уровень начальной подготовки поступающих студентов.* Студенты разных институтов (часто даже одного института), имеют различный уровень подготовленности. В результате, факультеты информатики вынуждены подгонять свои программы под уровень общей массы студентов.

- *Доступные ресурсы факультета.* Количество преподавателей на факультете информатики может варьироваться от трех-четырех в малых колледжах и гуманитарных учебных заведениях до 40 – 50 в больших исследовательских университетах. Очевидно, что малые факультеты не имеют достаточной гибкости. Таким факультетам необходимо разрабатывать набор приоритетов использования своих ограниченных ресурсов.

- *Интересы и практический опыт преподавателей.* Индивидуальные программы обучения варьируются в зависимости от специфических интересов и практических знаний преподавателей, особенно в малых институтах, где опыт сконцентрирован в узких областях.

Для разработки действительно удачного учебного плана необходимо найти баланс среди всех вышеприведенных факторов, что требует различных решений для разных учебных заведений. Не существует единой программы, одинаково подходящей всем. Каждый колледж и университет должен рассмотреть различные модели, предложенные в данном документе, и выработать учебный план, решающий свои конкретные задачи.

13.2. Руководящие принципы при разработке учебных курсов

Несмотря на тот факт, что разработка учебных курсов требует значительной локальной адаптации, можно указать некоторые ключевые принципы, которыми стоит руководствоваться в этом процессе:

Курс должен отражать целостность и характер информатики как независимой дисциплины. Информатика является независимой дисциплиной и характеризуется сочетанием теории, практики, знаний и навыков. Поэтому любой учебный курс в области информатики должен добиваться того, чтобы практические навыки, получаемые учащимися, основывались на теоретических знаниях и духе профессионализма.

Содержание учебного курса должно отвечать условиям быстрого технического прогресса и вырабатывать у студентов стремление к постоянному самосовершенствованию. Информатика является бурно развивающейся отраслью знания. Как уже обсуждалось в разделе 3, это приводит к необходимости регулярного обновления содержания учебных курсов. Не менее важно и то, что студенты должны научиться реагировать на быстрые изменения технологий и внешнего мира. Выпускники, специализирующиеся в области информатики, должны идти в

ногу с развитием отрасли и с интересом воспринимать новые знания. Одной из наиболее важных задач обучения информатики является воспитание специалистов, которые готовы учиться всю жизнь.

Разработка учебного курса должна руководствоваться целями, которые вы надеетесь достичь. Разрабатывая учебный курс, необходимо заранее наметить задачи этого курса, а также понять, какие знания появятся у студентов по окончании этого курса.. Эти цели, а также методы проверки их достижения, закладывают основу для всего курса. В США и других странах органы аккредитации учебных учреждений уделяют все большее внимание определению целей и стратегиям оценок знаний. Учебные программы, претендующие на актуальность, должны продемонстрировать соответствие учебных планов целям, которые они намереваются достичь.

Учебная программа в целом должна быть построена в духе новаторства, созидательности и профессионализма. Студенты лучше соответствуют предъявляемым к ним требованиям, если они понимают эти требования. Неправильно приучать студентов к определенной модели поведения в начальных курсах, и требовать сменить эту модель в более поздних курсах. На протяжении всего курса должно приветствоваться проявление студентами инициативы и воображения, желание освоить предмет в объеме, превышающем минимальные требования. В то же время, с самого начало надо поддерживать в студентах профессиональное и ответственное отношение к работе.

Необходимо обеспечить доступность учебного курса для широкого круга студентов. Очень часто учебные программы привлекают слишком однородную группу студентов, включающую недостаточное количество женщин или людей, чье материальное или социальное положение, а также этические взгляды отличаются от большинства. Хотя многие из причин, которые приводят к этому дисбалансу, следует искать за пределами университета, каждое учебное заведение должно стремиться к большему разнообразию своих программ, устраняя необъективность в учебных курсах и активно привлекая более широкий круг студентов.

Учебный курс должен дать студентам опыт, с помощью которого они смогут применить свои навыки и знания для решения практических проблем. Каждый студент-дипломник должен выполнить заключительный годовой проект, который потребует от студента использовать широкий диапазон практических навыков для решения сложной задачи. Некоторым аспектам информатики невозможно научить на лекции в аудитории. Такие навыки можно получить только самостоятельно решая практические задачи.

13.3. Необходимость соответствующих вычислительных мощностей

Высшее образование, конечно, зачастую страдает из-за недостаточного количества различных ресурсов. В известной степени, все образовательные программы должны принимать во внимание материальный фактор, ведь именно из-за финансовых ограничений обычно не удается реализовать все идеи по улучшению обучения.. Во многих отношениях эти ограничения в области информатики менее заметны, чем в других академических областях. Например, в настоящее время вычислительная техника доступна образовательным учреждениям, тогда как на заре преподавания информатики она, в основном, находилась преимущественно вне их досягаемости. В течение последних двадцати лет компьютеры стали товаром массового потребления, что сделало их намного более доступными.

В то же время, для образовательных учреждений важно принимать во внимание стоимость вычислительной техники. Сегодня эта стоимость складывается не только из стоимости аппаратного обеспечения – программы также представляют значительную долю расходов на вычислительную технику, особенно с учетом расходов на разработку курсов. Кроме того, необходимо учитывать расходы на сопровождение вычислительной техники, а также на заработную плату для обслуживающего персонала. Таким образом, учебная программа по информатике не может быть успешной без соответствующего финансирования вычислительных потребностей – как факультета, так и студентов.

За последние годы информатика стала, подобно биологии, химии и физике, лабораторной наукой. Изучение многих учебных курсов по информатике подразумевает выполнение запланированных и формализованных лабораторных работ. Эта практическая составляющая обучения приводит к возрастающей потребности в персонале для разработки материалов к курсам и проведения лабораторных занятий. Это также вносит свой вклад в стоимость финансовой поддержки высококачественной учебной программы по информатике. В определенной степени стоимость обучающих программ и других академических ресурсов может быть уменьшена за счет огромного количества ресурсов, доступных в World Wide Web. Список основных существующих в настоящее время ресурсов обучающих программ поддерживается на домашней странице ACM Special Interest Group in Computer Science Education (SIGCSE) по адресу http://www.acm.org/sigcse/.

13.4. Привлечение на работу квалифицированных сотрудников

Одна из наиболее сложных проблем, стоящих перед факультетами информатики – это найм на работу квалифицированных сотрудников. В большинстве академических областей число кандидатов значительно превышает

число вакансий. В информатике ситуация зачастую противоположная [Myers98, Roberts99], хотя имеются основания полагать, что данный кризис теряет свою остроту из-за уменьшения числа студентов, вызванного экономическим спадом. Недостаток кандидатов в сочетании с тем фактом, что ученые в области информатики получают высокие зарплаты вне академической сферы, делает затруднительным привлечение людей на работу и последующее их удержание.

Чтобы преодолеть эти трудности, мы рекомендуем образовательным учреждениям придерживаться следующей стратегии:

Принять агрессивный план набора персонала. Редкость какого-либо ресурса сама по себе не является причиной для прекращения поиска; дефицит кандидатов просто означает, что факультеты информатики должны искать их более настойчиво. Успех обычно является плодом инициативы и упорства. Факультеты должны начинать процесс набора людей на работу как можно раньше и рассматривать широкий круг потенциальных кандидатов, включая иностранных студентов и людей, работающих в индустрии.

Создать академические должности, сконцентрированные на преподавании. Как и в большинстве других областей, должности на факультете информатики обычно требуют степени кандидата наук (Ph.D.) и включают в себя как исследовательскую, так и преподавательскую деятельность. Если бы имелось достаточное количество кандидатов, обладающих требуемыми дипломами и навыками, было бы несложно настаивать на соответствующей квалификации претендентов. Однако в условиях недостатка кандидатов непонятно, могут ли факультеты информатики позволить себе такую избирательность. Далеко не каждое учебное заведение должно проводить полномасштабные исследования в области информатики. В то же время, сложно представить современный университет, который не предлагает никаких учебных курсов в этой области. Поэтому число возможных кандидатов можно увеличить за счет людей, которые имеют талант преподавателя, но не склонны к исследовательской работе.

Обеспечить преподавательскому составу поддержку, достаточную для того, чтобы люди не уходили в индустрию. Исследования, проведенные National Science Foundation в 1980-х годах, показали, что преподаватели, которые оставили университеты и ушли в индустрию, считали основной причиной своего ухода не экономические условия [Curtis83], а другие факторы. Среди причин назывался целый ряд особенностей, присущих преподавательской деятельности в университете – большие размеры групп студентов, тяжелое расписание, неадекватная поддержка исследовательской работы, отсутствие уверенности в завтрашнем дне и бюрократические препоны – все то, что названо в исследовании NSF "институционными препятствиями" (institutional disincentives). С ростом посещаемости курсов информатики задача обеспечения разумной загрузки персонала факультета приобретает весьма важное значение для учебного заведения.

Привлекать студентов выпускных курсов в качестве ассистентов. Кризис в области преподавания информатики возник из-за того, что слишком мало преподавателей должно удовлетворять потребности слишком большого числа студентов. Один из лучших способов бороться с этой проблемой – вовлекать в преподавательский процесс студентов выпускных курсов. Использование выпускников в качестве ассистентов не только помогает преодолеть нехватку преподавателей, но и дает ценный опыт преподавания самим выпускникам [Roberts95].

13.5. Заключение

Не существует универсального подхода к составлению хорошего учебного плана по информатике. Хотя мы полагаем, что рекомендации и предложения, приведенные здесь, окажутся полезными для широкого круга учебных заведений, каждое учебное заведение должно адаптировать эти рекомендации к своим условиям. Кроме того, важно критически подходить к учебным программам и регулярно обновлять их в соответствии с изменениями в предметной области. Современная учебная программа по информатике – это плод многих лет экспериментов и улучшений, предложенных преподавателями информатики в различных учебных заведениях. Учебная программа будущего во многом будет зависеть от того, насколько творчески подойдут ее авторы к применению положений этого отчета для разработки учебных курсов для студентов во всем мире.

БЛАГОДАРНОСТИ

Многие люди внесли свой вклад в проект CC2001 с момента его начала в конце 1998 года. Ниже приведен список тех, кто принимал участие, по крайней мере, в одной из рабочих групп: Ishfaq Ahmad, Robert Aiken, Anne Applin, Richard H. Austing, Scott Badman, Donald J. Bagert, Bruce Barnes, Mordechai (Moti) Ben-Ari, Julia Benson, Russell C. Bjork, Kevin W. Bowyer, Kim Bruce, Amy S. Bruckman, Bob Campbell, James Caristi, Doris Carver, Carl K. Chang, Morris Chang, Yiming Chen, Ashraful Chowdhury, Alan Clements, C. Fay Cover, Thad Crews, George T. Crocker, James H. Cross II, Steve Cunningham, Nell Dale, Andrea Danyluk, Gordon Davies, Susan Dean, Thomas G. Dietterich, John P. Dougherty, Sarah Douglas, J. Philip East, Dick Eckhouse, Gerald Engel, Edward Feigenbaum, Sue Fitzgerald, Ken Ford, Edward A. Fox, Josephine Freedman, Jose Galaviz, Dick Gayler, Benjamin Goldberg, Dina Golden, Don Gotterbarn, Saul Greenberg, Mark Guzdial, Cindy Hanchey, Elizabeth Hawthorne, Pat Hayes, Chris Haynes, Xudong He, Jim Hendler, Tom Hilburn, Wayne Horn, Cay Horstmann, Chuck Huff, Joseph Hummel, Phillip Hutto, John Impagliazzo, Michel Israel, Robert Jacob, Anil Jain, Carol Janik, Barbara Jennings, Ricardo Jiménez-Peris, Keith Jolly, Rhys Price Jones, Ioannis Kakadiaris, Willis King, Karl Klee, Timothy Klingler, Peter Knoke, Richard E. Korf, Norbert Kubilus, Amruth Kumar, Francis Lau, Gary Leavens, J. A. N. Lee, Ernst Leiss, James Lin, Cheng-Wen Liu, Ming T. (Mike) Liu, Tim Long, Philip Machanick, Raghu Machiraj, Raghu Machiraju, John Mallozzi, Bill Marion, C. Dianne Martin, Marta Patiño Martínez, Bruce R. Maxim, W. Michael McCracken, Chris McDonald, Andrew McGettrick, Susan Mengel, Dan Myers, John Mitchell, Michael Murphy, Walid Najjar, Thomas L. Naps, Patricia Nettnin, Gary Nutt, Yale Patt, Holly Patterson-McNeill, Richard E. Pattis, T. S. Pennington, Judy Porter, Jenny Preece, Anne-Louise Radimsky, Brad Richards, Eric Roberts, Ingrid Russell, Sartaj Sahni, Ahmed Sameh, Carolyn Schauble, G. Michael Schneider, Henning Schulzrinne, Russ Shackelford, Alfred Shin, Charles Shipley, Ben Shneiderman, Shai Simonson, Robert Sloan, Carl Smith, Milan Sonka, Sylvia Sorkin, Pradip Srimani, Lynn Andrea Stein, George Stockman, Devika Subramanian, Bobby Thrash, D. Singh Tomer, Frank Tong, Marilyn Mantei Tremaine, Alan Underwood, Ron Vetter, Henry Walker, David Waltz, Wenping Wang, Yun Wang, Tony Wasserman, Laurie Honour Werth, Curt M. White, Ed Wilkens, Barry Wilkinson, Terry Winograd, Ursula Wolz, и Anita Wright.

Мы также хотим поблагодарить многих людей, которые посещали различные семинары CC2001 в течение последних двух лет за их предложения и замечания. Отзывы, полученные во время этих встреч, оказали сильное влияние на структуру и границы этого документа.

Наконец, мы чрезвычайно признательны ACM, Компьютерному Обществу IEEE и National Science Foundation за оказанную поддержку.

БИБЛИОГРАФИЯ

[Abelson85] Harold Abelson and Gerald Jay Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press, 1985.

[ABET2000] Accreditation Board for Engineering and Technology. Accreditation policy and procedure manual. Baltimore, MD: ABET, Inc., November 2000. http://www.abet.org/images/policies.pdf.

[ACM65] ACM Curriculum Committee on Computer Science. An undergraduate program in computer science – preliminary recommendations. Communications of the ACM, 8(9):543-552, September 1965.

[ACM68] ACM Curriculum Committee on Computer Science. Curriculum '68: Recommendations for the undergraduate program in computer science. Communications of the ACM, 11(3):151-197, March 1968.

[ACM78] ACM Curriculum Committee on Computer Science. Curriculum '78: Recommendations for the undergraduate program in computer science. Communications of the ACM, 22(3):147-166, March 1979.

[ACM99] ACM Two-Year College Education Committee. Guidelines for associate degree and certificate programs to support computing in a networked environment. New York: The Association for Computing Machinery, September 1999.

[ACM2001] Association for Computing Machinery. ACM code of ethics and professional conduct. New York: The Association for Computing Machinery, May 2001. http://www.acm.org/constitution/code.html.

[AP2000] Advanced Placement Program. Introduction of Java in 2003-2004. The College Board. http://www.collegeboard.org/ap/computer-science December 20, 2000.

[BCS89a] British Computer Society and The Institution of Electrical Engineers. Undergraduate curricula for software engineers. London, June 1989.

[BCS89b] British Computer Society and The Institution of Electrical Engineers. Software in safety-related systems. London, October 1989.

[Beidler85] John Beidler, Richard Austing, and Lillian Cassel. Computing programs in small colleges. Communications of the ACM, 28(6):605-611, June 1985.

[Bennett86] W. Bennett. A position paper on guidelines for electrical and computer engineering education. IEEE Transactions in Education, E-29(3):175-177, August 1986.

[Bott91] Frank Bott, Allison Coleman, Jack Eaton, and Diane Rowland. Professional issues in software engineering. London: Pitman, 1991.

[Carnegie92] Carnegie Commission on Science, Technology, and Government. Enabling the future: Linking science and technology to societal goals. New York: Carnegie Commission, September 1992.

[COSINE67] COSINE Committee. Computer science in electrical engineering. Washington, DC: Commission on Engineering Education, September 1967.

[CSAB86] Computing Sciences Accreditation Board. Defining the computing sciences professions. October 1986. http://www.csab.org/comp_sci_profession.html.

[CSAB2000] Computing Sciences Accreditation Board. Criteria for accrediting programs in computer science in the United States. Version 1.0, January 2000. http://www.csab.org/criteria2k_v10.html.

[CSTB94] Computing Science and Telecommunications Board. Realizing the information future. Washington DC: National Academy Press, 1994.

[CSTB99] Computing Science and Telecommunications Board. Being fluent with information technology. Washington DC: National Academy Press, 1999.

[Curtis83] Kent K. Curtis. Computer manpower: Is there a crisis? Washington DC: National Science Foundation, 1983. http://www.acm.org/sigcse/papers/curtis83/.

[Davis97] Gordon B. Davis, John T. Gorgone, J. Daniel Couger, David L. Feinstein, and Herbert E. Longnecker, Jr. IS'97 model curriculum and guidelines for undergraduate degree programs in information systems. Association of Information Technology Professionals, 1997. http://webfoot.csom.umn.edu/faculty/gdavis/curcomre.pdf.

[Denning89] Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen B. Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. Communications of the ACM, 32(1):9-23, January 1989.

[Denning98] Peter J. Denning. Computing the profession. Educom Review, November 1998.

[Denning99] Peter J. Denning. Our seed corn is growing in the commons. http://www.cisp.org/imp/march_99/denning/03_99denning.htm Information Impacts Magazine, March 1999..

[EAB83] Educational Activities Board. The 1983 model program in computer science and engineering. Technical Report 932, Computer Society of the IEEE, December 1983.

[EAB86] Educational Activities Board. Design education in computer science and engineering. Technical Report 971, Computer Society of the IEEE, October 1986.

[EC77] Education Committee of the IEEE Computer Society. A curriculum in computer science and engineering. Publication EHO119-8, Computer Society of the IEEE, January 1977.

[Gibbs86] Norman E. Gibbs and Allen B. Tucker. Model curriculum for a liberal arts degree in computer science. Communications of the ACM, 29(3):202-210, March 1986.

[Gorgone2000] John T. Gorgone, Paul Gray, David L. Feinstein, George M. Kasper, Jerry N. Luftman, Edward A. Stohr, Joseph S. Valacich, and Rolf T. Wigand. MSIS 2000: Model curriculum and guidelines for graduate degree programs in information systems. ACM and AIS. http://cis.bentley.edu/ISA/pages/documents/msis2000jan00.pdf. January 2000.

[IEEE2001] Institute for Electrical and Electronic Engineers. IEEE code of ethics. Piscataway, USA. http://www.ieee.org/about/whatis/code.html. May 2001.
[Kelemen99] Charles F. Kelemen (editor), Owen Astrachan, Doug Baldwin, Kim Bruce, Peter Henderson, Dale Skrien, Allen Tucker, and Charles Ban Loan. Computer Science Report to the CUPM Curriculum Foundations Workshop in Physics and Computer Science. Report from a workshop at Bowdoin College, October 28-31, 1999.
[Koffman84] Elliot P. Koffman, Philip L. Miller, and Caroline E. Wardle. Recommended curriculum for CS1: 1984 a report of the ACM curriculum task force for CS1. Communications of the ACM, 27(10):998-1001, October 1984.
[Koffman85] Elliot P. Koffman, David Stemple, and Caroline E. Wardle. Recommended curriculum for CS2, 1984: A report of the ACM curriculum task force for CS2. Communications of the ACM, 28(8):815-818, August 1985.
[Lee98] Edward A. Lee and David G. Messerschmitt. Engineering and education for the future. IEEE Computer, 77-85, January 1998.
[Lidtke99] Doris K. Lidtke, Gordon E. Stokes, Jimmie Haines, and Michael C. Mulder. ISCC '99: An information systems-centric curriculum '99, July 1999. http://www.iscc.unomaha.edu.
[Martin96] C. Dianne Martin, Chuck Huff, Donald Gotterbarn, Keith Miller. Implementing a tenth strand in the CS curriculum. Communications of the ACM, 39(12):75-84, December 1996.
[Mulder75] Michael C. Mulder. Model curricula for four-year computer science and engineering programs: Bridging the tar pit. Computer, 8(12):28-33, December 1975.
[Mulder84] Michael C. Mulder and John Dalphin. Computer science program requirements and accreditation—an interim report of the ACM/IEEE Computer Society joint task force. Communications of the ACM, 27(4):330-335, April 1984.
[Mulder98] Fred Mulder and Tom van Weert. Informatics in higher education: Views on informatics and noninformatics curricula. Proceedings of the IFIP/WG3.2 Working Conference on Informatics (computer science) as a discipline and in other disciplines: What is in common? London: Chapman and Hall, 1998.
[Myers98] J. Paul Myers, Jr. and Henry M. Walker. The state of academic hiring in computer science: An interim review. SIGCSE Bulletin, 30(4):32a-35a, December 1998.
[NACE2001] National Association of Colleges and Employers. Job outlook '01 (online version). http://www.jobweb.com
[Neumann95] Peter G. Neumann. Computer related risks. New York: ACM Press, 1995.
[NSF96] National Science Foundation Advisory Committee. Shaping the future: New expectations for undergraduate education in science, mathematics, engineering, and technology. Washington DC: National Science Foundation, 1996.
[NTIA99] National Telecommunications and Information Administration. Falling through the Net: Defining the digital divide. Washington, DC: Department of Commerce, November 1999.
[Nunamaker82] Jay F. Nunamaker, Jr., J. Daniel Couger, Gordon B. Davis. Information systems curriculum recommendations for the 80s: Undergraduate and graduate programs. Communications of the ACM, 25(11):781-805, November 1982.
[OTA88] Office of Technology Assessment. Educating scientists and engineers: Grade school to grad school. OTA-SET-377. Washington, DC: U.S. Government Printing Office, June 1988.
[Paulk95] Mark Paulk, Bill Curtis, Mary Beth Chrissis, and Charles Weber. The capability maturity model: Guidelines for improving the software process. Reading, MA: Addison-Wesley, 1995.
[QAA2000] Quality Assurance Agency for Higher Education. A report on benchmark levels for computing. Gloucester, England: Southgate House, 2000.
[Ralston80] Anthony Ralston and Mary Shaw. Curriculum '78—Is computer science really that unmathematical. Communications of the ACM (23)2:67-70, February 1980.
[Roberts95] Eric Roberts, John Lilly, and Bryan Rollins. Using undergraduates as teaching assistants in introductory programming courses: An update on the Stanford experience. SIGCSE Bulletin (27)1:48-52, March 1995.
[Roberts99] Eric Roberts. Conserving the seed corn: Reflections on the academic hiring crisis. SIGCSE Bulletin (31)4:4-9, December 1999.
[SAC67] President's Science Advisory Commission. Computers in higher education. Washington DC: The White House, February 1967.
[SEEPP98] IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices (SEEPP). Software engineering code of ethics and professional practice (Version 5.2). http://www.acm.org/serving/se/code.htm.
[Shaw85] Mary Shaw. The Carnegie-Mellon curriculum for undergraduate computer science. New York: Springer-Verlag, 1985.
[Shaw91] Mary Shaw and James E Tomayko. Models for undergraduate courses in software engineering. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, January 1991.
[Shaw92] Mary Shaw. We can teach software better. Computing Research News 4(4):2-12, September 1992.
[SIGCHI92] Special Interest Group on Computer-Human Interaction. ACM SIGCHI Curricula for Human-Computer Interaction. New York: Association for Computing Machinery, 1992.
[SWEBOK01] Software Engineering Coordinating Committee. Guide to the Software Engineering Body of Knowledge (SWEBOK). Stone Man Version 0.95. A Project of the IEEE Computer Society, May 2001. http://www.swebok.org/stoneman/version095.html/.
[Tucker91] Allen B. Tucker, Bruce H. Barnes, Robert M. Aiken, Keith Barker, Kim B. Bruce, J. Thomas Cain, Susan E. Conry, Gerald L. Engel, Richard G. Epstein, Doris K. Lidtke, Michael C. Mulder, Jean B. Rogers, Eugene H. Spaf-

ford, and A. Joe Turner. Computing Curricula ’91. Association for Computing Machinery and the Computer Society of the Institute of Electrical and Electronics Engineers, 1991.

[Walker96] Henry M. Walker and G. Michael Schneider. A revised model curriculum for a liberal arts degree in computer science. Communications of the ACM, 39(12):85-95, December 1996.

[Zadeh68] Lofti A. Zadeh. Computer science as a discipline. Journal of Engineering Education, 58(8):913-916, April 1968.

ПРИЛОЖЕНИЕ А. СОВОКУПНОСТЬ ЗНАНИЙ ПО ИНФОРМАТИКЕ

Данное приложение к отчету CC2001 определяет область знаний, которая должна составлять основную часть учебных программ в области информатики. Обоснование такой классификации, а также подробная информация об истории, структуре и применении этой классификации приведены в полном отчете комиссии. Поскольку мы предполагаем, что приложения к отчету получат более широкое распространение, чем сам отчет, комитет считает важным включение в каждое приложение краткой сводки фундаментальных концепций, необходимых для понимания приведенных рекомендаций. Наиболее важные понятия описаны ниже в соответствующих разделах.

Структура совокупности знаний

Совокупность знаний по информатике организована в виде трехуровневой иерархической структуры. На верхнем уровне иерархии находится **область**, представляющая собой отдельную часть дисциплины информатики. Каждая область обозначается двухбуквенной аббревиатурой, например, OS для *операционных систем* или PL для *языков программирования.* Области делятся на меньшие структуры, называемые **разделами**, которые представляют собой отдельные тематические модули внутри области. Каждый раздел обозначается численным суффиксом, добавляемым к имени области, например, OS3 обозначает раздел *параллелизма*. Каждый раздел, в свою очередь, состоит из набора **тем**, представляющих собой нижний уровень этой иерархии.

Основные и факультативные разделы

Во время обновления данного раздела документа по сравнению с CC1991, комиссия была вынуждена принять во внимание тот факт, что информатика как дисциплина настолько расширилась, что студенты уже не могут освоить все темы, которые когда-либо считались фундаментальными. Поэтому комиссия решила определить минимальный набор **обязательных** курсов, включающий в себя только тот материал, который практически все преподаватели информатики признают необходимым для студентов, желающих получить диплом в области информатики. Материал, выходящий за рамки данного набора, рассматривается как **факультативный** (разделы по выбору). Настаивая на максимально распространенном определении основного набора знаний, комиссия надеется сохранить этот набор как можно более малым, давая тем самым образовательным учреждениям свободу в выборе факультативных компонент учебной программы с учетом их индивидуальных потребностей.

Обсуждая рекомендации CC2001 в процессе их разработки, мы пришли к выводу, что полезно подчеркнуть следующие соображения:

Обязательные курсы сами по себе не являются полной учебной программой. Поскольку набор обязательных курсов по определению является минимальным, он не может считаться полной учебной программой. Любая учебная программа должна включать факультативные разделы совокупности знаний, хотя содержание этих разделов и не определяется в данном документе.

Обязательные разделы не обязаны ограничиваться набором вводных курсов, читаемых на ранних стадиях учебной программы. Хотя многие из обязательных разделов действительно являются вводными по своему содержанию, имеется также несколько обязательных разделов, требующих для своего освоения солидный объем предварительных знаний. Например, комиссия полагает, что на каком-то этапе обучения все студенты должны самостоятельно разработать сложное приложение. Поэтому обязательный материал включает в себя сведения по управлению проектами – эти знания должны получить все студенты. Однако обычно такой проект выполняется ближе к концу обучения. Сходным образом, вводные курсы могут включать факультативные разделы, примыкающие к материалу курсов. Таким образом, термин *"обязательный курс"* ничего не говорит о периоде обучения, на котором этот курс может читаться.

Оценка времени, необходимого для изучения раздела

Для того чтобы дать читателям представление о времени, необходимом для изучения отдельного раздела, документ CC2001 определяет стандартные метрики. Выбор такой метрики оказался сложной задачей, так как не существует общепринятой меры этой величины. Для согласования с ранними версиями документа, комиссия решила измерять время в часах, что соответствует аудиторным часам, необходимым для представления материала в традиционном формате, ориентированном на лекции. Во избежание непонимания, однако, важно подчеркнуть следующие наблюдения, касающиеся нашего выбора единиц измерения:

Комиссия не ставит своей задачей рекомендовать лекционный формат. Хотя мы использовали метрику, основанную на классическом, лекционном стиле, комиссия уверена, что существуют другие методы, которые являются, по меньшей мере, столь же эффективными. Для многих из этих методов понятие *учебного часа* может оказаться не вполне адекватным. Но даже в этом случае временные характеристики могут послужить хотя бы в качестве меры сравнения, в том смысле, что 5-часовой раздел будет предположительно занимать в пять раз больше времени, чем 1-часовой, независимо от стиля преподавания.

Указываемые часы не включают в себя время, проводимое вне аудитории. Время, отводимое на раздел, не включает в себя время подготовки преподавателя и время, затрачиваемое студентами вне аудитории. В качестве рекомендации заметим, что объем внеаудиторных занятий должен примерно в три раза превосходить объем аудиторных. Так, раздел, требующий 3 часа должен обычно изучаться 12 часов (3 часа в аудитории и 9 часов самостоятельно).

Указываемые часы, отводимые на раздел, подразумевают минимальный объем сведений. Временные показатели, отведенные нами для каждого раздела, должны пониматься как минимальное количество времени, требуемое студенту для освоения раздела в рамках, требуемых программой. Всегда допустимо и полезно отводить на раздел больше времени, чем обязательный минимум.

Совокупность знаний по информатике

Краткий обзор совокупности знаний по информатике с указанием областей, разделов, основных разделов и минимального времени, требуемого для каждого раздела, приведен на рисунке А-1. Подробное описание каждой области дано ниже в соответствующих разделах.

Рисунок А-1. Совокупность знаний по информатике (обязательные темы подчеркнуты)

DS. Дискретные структуры (43 часа)
DS1. Функции, отношения и множества (6)
DS2. Основы логики (10)
DS3. Методы доказательства (12)
DS4. Основы вычислений (5)
DS5. Графы и деревья (4)
DS6. Дискретная вероятность (6)

PF. Основы программирования (38 часов)
PF1. Основные конструкции программирования (9)
PF2. Алгоритмы и решение задач (6)
PF3. Фундаментальные структуры данных (14)
PF4. Рекурсия (5)
PF5. Событийно-управляемое программирование (4)

AL. Алгоритмы и теория сложности (31 час)
AL1. Основы анализа алгоритмов (4)
AL2. Алгоритмические стратегии (6)
AL3. Фундаментальные вычислительные алгоритмы (12)
AL4. Распределенные алгоритмы (3)
AL5. Основы теории вычислимости (6)
AL6. Классы сложности P и NP
AL7. Теория автоматов
AL8. Углубленный анализ алгоритмов
AL9. Криптографические алгоритмы
AL10. Геометрические алгоритмы
AL11. Параллельные алгоритмы

AR. Архитектура и организация ЭВМ (36 часов)
AR1. Цифровая логика и цифровые системы (6)
AR2. Представление данных в памяти компьютера (3)
AR3. Организация машины на уровне ассемблера (9)
AR4. Устройство памяти компьютера (5)
AR5. Взаимодействие и коммуникации (3)
AR6. Функциональная организация (7)
AR7. Многопроцессорные и альтернативные архитектуры (3)
AR8. Улучшение производительности
AR9. Архитектура сетевых и распределенных систем

OS. Операционные системы (18 часов)
OS1. Обзор операционных систем (2)
OS2. Основы операционных систем (2)
OS3. Параллелизм (6)
OS4. Планирование и диспетчеризация (3)
OS5. Управление памятью (5)
OS6. Управление устройствами
OS7. Безопасность и защита данных
OS8. Файловые системы
OS9. Встроенные системы и системы реального времени
OS10. Отказоустойчивость
OS11. Оценка производительности системы
OS12. Языки сценариев

NC. Распределенные вычисления (15 часов)
NC1. Введение в распределенные вычисления (2)
NC2. Сети и телекоммуникации (7)
NC3. Сетевая безопасность (3)
NC4. Web как пример архитектуры "клиент-сервер" (3)
NC5. Разработка web-приложений
NC6. Управление сетями
NC7. Сжатие и распаковка данных
NC8. Технологии мультимедиа
NC9. Беспроводные и мобильные компьютеры

PL. Языки программирования (21 час)
PL1. Обзор языков программирования (2)
PL2. Виртуальные машины (1)
PL3. Введение в трансляцию (2)
PL4. Переменные и типы данных (3)
PL5. Механизмы абстракции (3)
PL6. Объектно-ориентированное программирование (10)
PL7. Функциональное программирование
PL8. Системы трансляции
PL9. Системы типов
PL10. Семантика языков программирования
PL11. Разработка языков программирования

HC. Взаимодействие человека и машины (8 часов)
HC1. Основы взаимодействия человека и машины (6)
HC2. Построение простого графического интерфейса (2)
HC3. Оценка программного обеспечения, ориентированного на человека
HC4. Разработка программного обеспечения, ориентированного на человека
HC5. Проектирование графического интерфейса пользователя
HC6. Программирование графического интерфейса пользователя
HC7. Человеко-машинные аспекты мультимедиа-систем
HC8. Человеко-машинные аспекты сотрудничества и коммуникаций

GV. Компьютерная графика и визуализация (3 часа)
GV1. Фундаментальные методы в графике (2)
GV2. Графические системы (1)
GV3. Графические коммуникации
GV4. Геометрическое моделирование
GV5. Основы рендеринга
GV6. Углубленное изучение рендеринга
GV7. Более сложные методы
GV8. Компьютерная анимация
GV9. Визуализация
GV10. Виртуальная реальность
GV11. Компьютерное зрение

IS. Интеллектуальные системы (10 часов)
IS1. Основные вопросы, связанные с интеллектуальными системами (1)
IS2. Поиск решений (5)
IS3. Представление знаний и вывод (4)
IS4. Углубленное изучение поиска
IS5. Углубленное изучение представления знаний и вывода
IS6. Агенты
IS7. Обработка естественного языка
IS8. Обучение машины и нейронные сети
IS9. Системы искусственного интеллекта с планируемым поведением
IS10. Робототехника

IM. Управление информацией (10 часов)
IM1. Информационные модели и системы (3)
IM2. Системы баз данных (3)
IM3. Моделирование данных (4)
IM4. Реляционные базы данных
IM5. Языки запросов к базам данных
IM6. Проектирование реляционных баз данных
IM7. Обработка транзакций
IM8. Распределенные базы данных
IM9. Проектирование физической структуры базы данных
IM10. Извлечение информации
IM11. Хранение и поиск информации
IM12. Гипертекст и гипермедиа
IM13. Мультимедийная информация и системы мультимедиа
IM14. Цифровые библиотеки

SP. Социальные и профессиональные вопросы (16 часов)
SP1. История информатики (1)
SP2. Социальный контекст информатики (3)
SP3. Методы и средства анализа (2)
SP4. Профессиональная и этическая ответственность (3)
SP5. Недостатки компьютерных систем и риски, связанные с их применением (2)
SP6. Интеллектуальная собственность (3)
SP7. Конфиденциальность и гражданские свободы (2)
SP8. Компьютерные преступления
SP9. Экономические вопросы, связанные с применением компьютеров
SP10. Философские концепции

SE. Программная инженерия (31 час)
SE1. Проектирование ПО (8)
SE2. Использование программных интерфейсов приложений (5)
SE3. Программные средства и окружения (3)
SE4. Процессы разработки ПО (2)
SE5. Спецификации и требования к ПО (4)
SE6. Проверка соответствия ПО (3)
SE7. Эволюция ПО (3)
SE8. Управление программными проектами (3)
SE9. Компонентно-ориентированная разработка
SE10. Формальные методы
SE11. Надежность ПО
SE12. Разработка специализированных систем

CN. Вычислительная математика и численные методы (нет обязательных часов)
CN1. Численный анализ
CN2. Исследование операций
CN3. Моделирование
CN4. Высокопроизводительные вычисления

Замечание: Числа в скобках обозначают минимальное число часов, необходимое для изучения материала в лекционном формате. Всегда допустимо отводить больше времени.

Дискретные структуры (DS)

DS1. Функции, отношения и множества [обязательный]
DS2. Основы логики [обязательный]
DS3. Методы доказательства [обязательный]
DS4. Основы вычислений [обязательный]
DS5. Графы и деревья [обязательный]
DS6. Дискретная вероятность [обязательный]

Дискретные структуры (discrete structures) являются фундаментальной основой информатики. Говоря *фундаментальная*, мы подразумеваем, что сравнительно небольшое число ученых будут непосредственно работать в данной области, но при этом многие другие разделы информатики требуют умения работать с концепциями дискретных структур. Дискретные структуры включают важный материал из таких областей, как теория множеств, логика, теория графов и комбинаторика.

Сведения из теории дискретных структур широко используются не только в структурах данных и алгоритмах, но и во всех остальных разделах информатики. Например, при проверке формальных спецификаций, верификации, а также в криптографии необходимо уметь создавать и понимать формальные доказательства. Понятия теории графов используются в сетях, операционных системах и компиляторах. Теория множеств находит применение в программной инженерии и базах данных.

По мере развития информатики, все более и более сложные методы анализа оказывают влияние на практические проблемы. Для того, чтобы освоить вычислительные средства будущего, сегодняшним студентам потребуется твердое знание дискретных структур.

В заключение заметим, что существуют области знания, границы которых очень трудно определить, и теория дискретных структур, безусловно, является одним из примеров таких областей. Здесь собраны математические основы, которые должны преподаваться при обучении информатики, и которые достаточно хорошо известны, чтобы преподаватели информатики могли читать их с большой степенью подробности. Однако, решение о том, где проходит граница между темами, освещаемыми в дискретных структурах, алгоритмах или теории сложности, с одной стороны, и темами, оставленными в виде вспомогательных разделов математики, с другой стороны, неизбежно носит несколько волюнтаристский характер. Мы напоминаем читателям, что в обеих областях есть темы, которые могут быть включены в некоторые университетские программы под названием "дискретные структуры".

DS1. Функции, отношения и множества [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Функции (сюръекции, инъекции, обратные функции, композиция)
Отношения (рефлексивность, симметричность, транзитивность, эквивалентность)
Множества (диаграммы Венна, дополнения, декартовы произведения, степенные множества)
Принцип Дирихле
Мощность и счетность

Задачи обучения:

1. Объяснить с примерами основы терминологии функций, отношений и множеств.
2. Обучить выполнению операций, связанных с множествами, функциями и отношениями.
3. Связать практические примеры с подходящими моделями множеств, функций и отношений, а также дать в этом контексте интерпретацию соответствующих операций.
4. Продемонстрировать основные принципы, включая использование диагонализации и принципа Дирихле.

DS2. Основы логики [обязательный]

Минимальное время, отводимое на раздел: 10 часов

Темы:

Логика высказываний
Логические связки
Таблицы истинности
Нормальные формы (конъюнктивные и дизъюнктивные)
Общезначимость (тавтология)
Логика предикатов
Кванторы всеобщности и существования
Правила modus ponens и modus tollens

Ограничения логики предикатов

Задачи обучения:
1. Обучить применению формальных методов символической логики высказываний и логики предикатов.
2. Показать использование формальных средств символической логики для моделирования алгоритмов и реальных жизненных ситуаций.
3. Использовать формальные логические доказательства и логическое рассуждение для решения задач, например, головоломок.
4. Описать применимость и ограничения логики предикатов.

DS3. Методы доказательства [обязательный]

Минимальное время, отводимое на раздел: 12 часов

Темы:

Понятия импликации, обращения, противопоставления, отрицания и противоречия
Структура формальных доказательств
Прямые доказательства
Доказательство через контрпример
Доказательство через противопоставление
Доказательство через противоречие
Математическая индукция
Сильная индукция
Рекурсивные математические определения
Вполне упорядоченные множества

Задачи обучения:
1. Обрисовать основную структуру и дать примеры каждого метода доказательств, описанных выше.
2. Обсудить, какой вид доказательства лучше подходит для данной задачи.
3. Связать идеи математической индукции с понятием рекурсии и рекурсивно определенных структур.
4. Указать различия между математической и сильной индукцией и дать примеры адекватного использования каждого из этих методов.

DS4. Основы вычислений [обязательный]

Минимальное время, отводимое на раздел: 5 часов

Темы:

Основы вычислений:
- Правила суммы и произведения
- Принцип включения-выключения
- Арифметические и геометрические прогрессии
- Числа Фибоначчи

Принцип Дирихле
Перестановки и сочетания
- Основные определения
- Тождество Паскаля
- Биномиальная теорема

Решение рекуррентных соотношений
- Общие примеры
- Основная теорема рекуррентных соотношений

Задачи обучения:
1. Научиться вычислять перестановки и сочетания множества, а также интерпретировать их значения в контексте конкретного приложения.
2. Сформулировать основную теорему рекуррентных соотношений.
3. Научиться решать типичные рекуррентные соотношения.
4. Научиться анализировать задачу для того, чтобы построить соответствующие рекуррентные уравнения или выявить связанные с ней вычислительные вопросы.

DS5. Графы и деревья [обязательный]

Минимальное время, отводимое на раздел: 4 часа

Темы:

Деревья
Неориентированные графы
Ориентированные графы
Остовные деревья
Стратегии обхода графов

Задачи обучения:

1. Проиллюстрировать на примерах основные понятия теории графов, а также их свойства и некоторые специальные случаи.
2. Продемонстрировать различные методы обхода деревьев и графов.
3. Дать примеры моделирования задач информатики с использованием деревьев и графов.
4. Показать связь графов и деревьев со структурами данных, алгоритмами и вычислениями.

DS6. Дискретная вероятность [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Конечное вероятностное пространство, вероятностная мера, события
Условная вероятность, независимость событий, теорема Байеса
Целочисленные случайные величины, математическое ожидание

Задачи обучения:

1. Научиться вычислять вероятности событий и математического ожидания случайных величин для элементарных задач, таких как игра в рулетку.
2. Усвоить различия между независимыми и зависимыми событиями.
3. Научиться применять биномиальную теорему для независимых событий и теорему Байеса для зависимых событий.
4. Научиться применять вероятностные методы к решению таких задач, как метод Монте-Карло, анализ среднего случая алгоритмов и хеширование.

Основы программирования (PF)

PF1. Основные конструкции программирования [обязательный]
PF2. Алгоритмы и решение задач [обязательный]
PF3. Фундаментальные структуры данных [обязательный]
PF4. Рекурсия [обязательный]
PF5. Событийно-управляемое программирование [обязательный]

Знание основ программирования (programming fundamentals) является необходимым условием для освоения большинства разделов информатики. В документе CC1991 знание языков программирование рассматривалось как обязательное, и все же ему уделялось недостаточно внимания. Раздел "Введение в языки программирования" в документе CC1991 рассматривался как факультативный, ему отводилось лишь 12 часов аудиторных занятий, что оправдывалось оптимистическими предположениями о том, что "все больше студентов … получают такие знания в средней школе". Мы уверены, что программы университетов по информатике должны учить студентов как грамотно использовать хотя бы один язык программирования. Более того, мы рекомендуем, чтобы учебные программы включали в себя освоение языков, которые используют, по крайней мере, две парадигмы программирования. На выполнение этого требуется значительно больше 12 часов.

Эта область знаний состоит из тех концепций и навыков, которые важны для практики программирования независимо от применяемой парадигмы программирования. Поэтому данный раздел включает в себя материал по фундаментальным концепциям программирования, основным структурам данных и алгоритмам. Этот материал, однако, никоим образом не покрывает весь спектр программистских знаний, которыми должен обладать студент, специализирующийся в области информатики. Многие другие области, наиболее заметными среди которых являются языки программирования (PL) и программное обеспечение (SE) также содержат обязательный материал, относящийся к программированию. В большинстве случаев, этот материал с тем же успехом можно было бы расположить в данном разделе или в более сложных разделах.

PF1. Основные конструкции программирования [обязательный]

Минимальное время, отводимое на раздел: 9 часов

Темы:

Основы синтаксиса и семантики языков высокого уровня
Переменные, типы, выражения и присваивания
Основы ввода/вывода
Операторы проверки условия и цикла
Функции и передача параметров
Структурная декомпозиция

Задачи обучения:

1. Проанализировать и объяснить поведение простых программ, включающих фундаментальные конструкции, рассматриваемые в данном разделе.
2. Модифицировать и расширить короткие программы, использующие стандартные условные и итеративные операторы и функции.
3. Спроектировать, реализовать, протестировать и отладить программу, которая использует все перечисленные ниже конструкции программирования: последовательное исполнение, простой ввод/вывод, стандартные условные операторы и циклы, определения функций.
4. Научиться выбирать подходящие условные операторы и циклы для данной задачи программирования.
5. Научиться применять методы структурной (функциональной) декомпозиции для разделения программы на части.
6. Описать механизм передачи параметров.

PF2. Алгоритмы и решение задач [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Стратегии решения задач
Роль алгоритмов в процессе решения задач
Стратегии реализации алгоритмов
Стратегии отладки
Концепции и свойства алгоритмов

Задачи обучения:

1. Обсудить важную роль алгоритмов в процессе решения задач.
2. Указать свойства, присущие хорошим алгоритмам.
3. Разработать алгоритмы для решения простых задач.
4. Использовать псевдокод или язык программирования для реализации, тестирования и отладки простых алгоритмов.
5. Описать стратегии, полезные при отладке.

PF3. Фундаментальные структуры данных [обязательный]

Минимальное время, отводимое на раздел: 14 часов

Темы:

Примитивные типы
Массивы
Записи
Строки и операции со строками
Представление данных в памяти
Статическое, автоматическое и динамическое выделение памяти
Управление памятью во время исполнения программы
Указатели и ссылки
Связные структуры
Методы реализации стеков, очередей и хэш-таблиц
Методы реализации графов и деревьев
Стратегии выбора подходящей структуры данных

Задачи обучения:

1. Обсудить представление и использование примитивных типов данных и встроенных структур данных.
2. Обсудить, как перечисленные структуры данных представляются в памяти.
3. Описать типичные применения каждой из перечисленных структур данных.
4. Реализовать структуры данных, определенные пользователем, на языке высокого уровня.
5. Сравнить альтернативные представления структур данных с точки зрения производительности.
6. Написать программы, которые используют каждую из следующих структур данных: массивы, записи, строки, связные списки, стек, очереди и хэш-таблицы.
7. Сравнить преимущества и накладные расходы динамической и статической реализации структур данных.
8. Научиться выбирать подходящие структуры данных для решения задачи.

PF4. Рекурсия [обязательный]

Минимальное время, отводимое на раздел: 5 часов

Темы:

Понятие рекурсии
Рекурсивные математические функции
Простые рекурсивные процедуры
Стратегия "разделяй и властвуй"
Рекурсивный перебор с возвратами
Реализация рекурсии

Задачи обучения:

1. Привести понятие рекурсии и примеры использования рекурсии.
2. Дать общую постановку рекурсивно определенной задачи.
3. Сравнить итеративные и рекурсивные решения элементарных задач, таких как вычисление факториала.
4. Описать стратегию "разделяй и властвуй".
5. Реализовать, протестировать и отладить простые рекурсивные функции и процедуры.
6. Показать, как рекурсия может быть реализована с использованием стека.
7. Обсудить, для каких классов задач алгоритмы типа "перебор с возвратом" являются хорошим решением.
8. Объяснить, в каких случаях рекурсия подходит для решения задач.

PF5. Событийно-управляемое программирование [обязательный]

Минимальное время, отводимое на раздел: 4 часа

Темы:

Методы обработки событий
Распространение событий
Обработка событий

Задачи обучения:

1. Объяснить различие между событийно-управляемым программированием и программированием в командной строке.
2. Спроектировать, запрограммировать, протестировать и отладить простые программы, реагирующие на события, инициируемые пользователем.
3. Разработать код, который обрабатывает исключительные ситуации, возникающие во время выполнения.

Алгоритмы и сложность (AL)

AL1. Основы анализа алгоритмов [обязательный]
AL2. Алгоритмические стратегии [обязательный]
AL3. Фундаментальные вычислительные алгоритмы [обязательный]
AL4. Распределенные алгоритмы [обязательный]
AL5. Основы теории вычислимости [обязательный]
AL6. Классы сложности P и NP [факультативный]
AL7. Теория автоматов [факультативный]
AL8. Углубленный анализ алгоритмов [факультативный]
AL9. Криптографические алгоритмы [факультативный]
AL10. Геометрические алгоритмы [факультативный]
AL11. Параллельные алгоритмы [факультативный]

Теория алгоритмов (algorithms and complexity) является основой информатики и программной инженерии. Фактическая производительность любой программной системы зависит от двух факторов: (1) применяемых в ней алгоритмов и (2) эффективности реализации на различных ее уровнях. Поэтому разработка хорошего алгоритма имеет решающее значение для производительности любой программной системы. Кроме того, изучение алгоритмов позволяет более глубоко вникнуть в задачу и может подсказать методы решения, не зависящие от языка программирования, парадигмы программирования, аппаратного обеспечения и других аспектов реализации.

Важной составной частью знаний в области информатики является способность выбирать алгоритм, подходящий для решения данной задачи, или доказать, что такого алгоритма не существует. Эта способность основывается на знании класса алгоритмов, которые предназначены для решения определенного набора известных задач, понимании их сильных и слабых сторон, применимости различных алгоритмов в данном контексте. Эффективность является важнейшим вопросом в данной области.

AL1. Основы анализа алгоритмов [обязательный]

Минимальное время, отводимое на раздел: 4 часа

Темы:

Асимптотический анализ поведения алгоритмов в среднем и крайних случаях
Различия между поведением в лучшем, среднем и худшем случае
Нотация: **O** большое, **o** малое, **омега**, и **тэта**
Стандартные классы сложности
Эмпирические измерения производительности
Компромисс между временем и объемом памяти в алгоритмах
Использование рекуррентных отношений для анализа рекурсивных алгоритмов

Задачи обучения:

1. Научиться использовать нотацию **O** большого, **омега** и **тэта** для описания объема вычислений, производимых алгоритмом.
2. Научиться использовать нотацию **O** большого, **омега** и **тэта** для описания асимптотических оценок снизу и сверху, а также точной границы.
3. Определить сложность по времени и памяти простых алгоритмов.
4. Вывести рекуррентные соотношения, описывающие временную сложность рекурсивно определенных алгоритмов.
5. Решить простые рекуррентные соотношения.

AL2. Алгоритмические стратегии [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Алгоритмы полного перебора
"Жадные" алгоритмы
Алгоритмы "разделяй и властвуй"
Перебор с возвратом
Метод ветвей и границ
Эвристики
Сопоставление с образцом и алгоритмы обработки текста
Алгоритмы численной аппроксимации

Задачи обучения:

1. Описать недостатки алгоритмов полного перебора.
2. Для каждого из следующих видов алгоритмов (полного перебора, "жадных", "разделяй и властвуй", ветвей и границ и эвристического), указать примеры из обыденной жизни, которые иллюстрируют основные понятия.
3. Найти задачу, для решения которой подходит "жадный" алгоритм, и реализовать его.
4. Найти задачу, для решения которой подходит алгоритм "разделяй и властвуй", и реализовать его.
5. Использовать перебор с возвратом для решения задачи поиска выхода из лабиринта.
6. Описать различные эвристические методы решения задач.
7. Использовать сопоставление с образцом для поиска подстроки в тексте.
8. Использовать численную аппроксимацию для решения математических задач, таких как нахождение корней многочлена.

AL3. Фундаментальные вычислительные алгоритмы [обязательный]

Минимальное время, отводимое на раздел: 12 часов

Темы:

Простые численные алгоритмы
Алгоритмы последовательного и бинарного поиска
Квадратичные методы сортировки (сортировка методом выбора, сортировка вставками)
Алгоритмы сортировки за время O(N log N) (быстрая сортировка, пирамидальная сортировка, сортировка слиянием)
Хэш-таблицы, включая методы уменьшения количества коллизий
Бинарные деревья поиска
Представление графов (списки смежности, матрица смежности)
Обходы в глубину и ширину
Алгоритмы поиска кратчайшего пути (алгоритмы Дейкстры и Флойда)
Транзитивное замыкание (алгоритм Флойда)
Минимальное остовное дерево (алгоритмы Прима и Крускала)
Топологическая сортировка

Задачи обучения:

1. Реализовать наиболее распространенные алгоритмы сортировки за квадратичное время и за время *O*(N log N).
2. Придумать и реализовать хорошую хэш-функцию для данной задачи.
3. Придумать и реализовать алгоритм разрешения коллизий в хэш-таблице.
4. Обсудить вычислительную сложность основных алгоритмов сортировки, поиска и хеширования.
5. Обсудить другие факторы, помимо вычислительной сложности, влияющие на выбор алгоритмов (время кодирования, сопровождаемость и использование знаний о типичных для данного приложения шаблонов входных данных).
6. Научиться решать задачи, используя фундаментальные алгоритмы на графах: поиск в глубину и в ширину, нахождение кратчайших путей от одного источника и между всеми узлами, транзитивное замыкание, топологическая сортировка и, по крайней мере, один вариант построения минимального остовного дерева.
7. Продемонстрировать следующие навыки: оценка алгоритмов, выбор алгоритма для решения данной задачи, оправдание выбора, реализация алгоритма.

AL4. Распределенные алгоритмы [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Алгоритмы консенсуса и голосования
Распознавание завершения
Устойчивость к отказам
Стабилизация

Задачи обучения:

1. Объяснить распределенную парадигму.
2. Объяснить один простой распределенный алгоритм.
3. Определить, когда следует использовать алгоритмы консенсуса и голосования.
4. Различать физические и логические часы.
5. Объяснить относительное упорядочение событий в распределенном алгоритме.

AL5. Основы теории вычислимости [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Конечные автоматы
Контекстно-свободные грамматики
Легко- и трудноразрешимые задачи
Невычислимые функции
Проблемы завершения
Следствия невычислимости

Задачи обучения:

1. Обсудить понятие конечного автомата.

2. Объяснить понятие контекстно-свободной грамматики.
3. Разработать детерминированный конечный автомат, распознающий заданный язык.
4. Объяснить, почему некоторые проблемы не имеют алгоритмического решения.
5. Дать примеры, иллюстрирующие понятие неразрешимости.

AL6. Классы сложности P и NP [факультативный]

Темы:

Определения классов P и NP
NP-полнота (теорема Кука)
Стандартные NP-полные задачи
Методы сведения

Задачи обучения:

1. Определить классы задач P и NP.
2. Объяснить значение NP-полноты.
3. Доказать NP-полноту задачи путем сведения к ней одной из классических NP-полных задач.

AL7. Теория автоматов [факультативный]

Темы:

Детерминированные конечные автоматы (DFA)
Недетерминированные конечные автоматы (NFA)
Эквивалентность DFA и NFA
Регулярные выражения
Лемма об условиях регулярности языка
Магазинные автоматы (PDA)
Связь между PDA и контекстно-свободными грамматиками
Свойства контекстно-свободных грамматик
Машины Тьюринга
Недетерминированные машины Тьюринга
Множества и языки
Иерархия Хомского
Тезис Черча-Тьюринга

Задачи обучения:

1. Определить положение языка в иерархии Хомского (автоматный, контекстно-свободный, контекстно-зависимый и рекурсивно-перечислимый).
2. Доказать, что язык относится к данному классу, и не принадлежит к следующему классу иерархии Хомского.
3. Научиться задавать язык с помощью одной из эквивалентных нотаций: DFA, NFA, регулярные выражения для автоматных языков, PDA и CFG – для контекстно-свободных.
4. Объяснить, по крайней мере, по одному алгоритму для нисходящего и восходящего разбора.
5. Объяснить тезис Черча-Тьюринга и его значение.

AL8. Углубленный анализ алгоритмов [факультативный]

Темы:

Амортизационный анализ
Online- и offline-алгоритмы
Рандомизированные алгоритмы
Динамическое программирование
Комбинаторная оптимизация

Задачи обучения:

1. Использовать метод потенциала для проведения амортизационного анализа незнакомой структуры данных с помощью заданной функции потенциала.
2. Объяснить, почему состязательный анализ подходит для разбора online-алгоритмов.
3. Объяснить использование рандомизации на примере разработки алгоритма для задачи, детерминированный алгоритм для которой неизвестен или намного сложнее, чем рандомизированный.
1. Придумать и реализовать решение для задачи методом динамического программирования.

AL9. Криптографические алгоритмы [факультативный]

Темы:

Исторический обзор криптографии
Криптография с секретным ключом и проблема обмена ключами
Криптография с открытым ключом
Цифровые подписи
Протоколы безопасности
Приложения (доказательства с нулевым разглашением, аутентификация и т.д.)

Задачи обучения:

1. Описать базовые алгоритмы теории чисел, включая нахождение наибольшего общего делителя, мультипликативное обращение по модулю n, возведение в степень n.
2. Описать хотя бы одну криптосистему с открытым ключом, включая необходимые обоснования ее безопасности с точки зрения теории сложности.
3. Создать простые расширения криптографических протоколов, используя известные протоколы и криптографические примитивы.

AL10. Геометрические алгоритмы [факультативный]

Темы:

Отрезки прямой: свойства, пересечения
Алгоритмы построения выпуклой оболочки

Задачи обучения:

1. Описать и дать анализ временной сложности по крайней мере двух алгоритмов построения выпуклой оболочки.
2. Доказать, что **Omega**(N log N) является нижней оценкой для алгоритмов построения выпуклой оболочки.
3. Описать, по крайней мере, еще один эффективный алгоритм вычислительной геометрии, например, нахождение пары ближайших точек, выпуклые уровни или максимальные уровни.

AL11. Параллельные алгоритмы [факультативный]

Темы:

Модель PRAM
Монопольные и параллельные операции ввода-вывода
Переходы по указателям
Теорема Брента и эффективность работы

Задачи обучения:

1. Описать реализацию связных списков в PRAM.
2. Использовать операции с параллельными префиксами для выполнения простых вычислений.
3. Объяснить теорему Брента и ее значение.

Архитектура и организация ЭВМ (AR)

AR1. Цифровая логика и цифровые системы [обязательный]
AR2. Представление данных в памяти компьютера [обязательный]
AR3. Организация машины на уровне ассемблера [обязательный]
AR4. Устройство памяти компьютера [обязательный]
AR5. Взаимодействие и коммуникации [обязательный]
AR6. Функциональная организация [обязательный]
AR7. Многопроцессорные и альтернативные архитектуры [обязательный]
AR8. Улучшение производительности [факультативный]
AR9. Архитектура сетевых и распределенных систем [факультативный]

Компьютер – основа вычислений. Без компьютера современные вычислительные дисциплины были бы всего лишь ветвью теоретической математики. Современный профессионал в любой области информатики не может воспринимать компьютер как черный ящик, исполняющий программы с помощью неизвестной магии. Все студенты, изучающие информатику, должны понимать устройство функциональных компонент, из которых состоит компьютер, их характеристики, производительность и взаимодействие между ними. Понимание архитектуры и организации компьютера (architecture and organization) также позволяет писать более эффективные программы.

При выборе используемой системы, они должны понимать значение характеристик различных компонентов, таких как тактовая частота процессора и объем памяти.

Темы данного раздела относятся преимущественно к обязательному материалу и рассчитаны на поддержку учебных программ, отводящих на изучение архитектуры компьютера только 36 часов. Для тех программ, которые рассчитывают количество часов, большее минимального, те же темы (AR1–AR7) можно изучать и более глубоко, посвятив им два последовательных курса. Для программ с акцентом на факультативные курсы, данный материал можно разделить на два курса и затем более подробно осветить в третьем курсе по выбору.

AR1. Цифровая логика и цифровые системы [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Обзор и история архитектуры компьютеров
Логические элементы компьютера (логические вентили, триггеры, счетчики, регистры, программируемая логическая матрица)
Логические выражения, минимизация, дизъюнкция конъюнкций
Нотация регистровых передач
Физические вопросы (вентильные задержки, нагрузочные модули по входу и выходу)

Задачи обучения:

1. Описать эволюцию компьютерных архитектур от электронных ламп до СБИС.
2. Продемонстрировать понимание логических элементов и их роли в историческом развитии компьютерных архитектур.
3. Научиться использовать математические выражения для описания функций простых последовательных и комбинационных схем.
4. Разработать простую схему, используя логические элементы.

AR2. Представление данных в памяти компьютера [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Биты, байты и слова
Представление числовых данных и системы счисления
Системы с фиксированной и плавающей точкой
Представление со знаковым битом и в дополнительном коде
Представление нечисловых данных (коды символов, графические данные)
Представление массивов и записей

Задачи обучения:

1. Объяснить причины использования различных форматов представления числовых данных.
2. Объяснить, как отрицательные целые числа хранятся в представлениях со знаковым битом и в дополнительном коде.
3. Научиться преобразовывать числовые данные из одного формата в другой.
4. Обсудить, как представление данных с фиксированной разрядностью влияет на точность.
5. Обсудить внутреннее представление нечисловых данных.
6. Обсудить внутреннее представление символов, строк, записей и массивов.

AR3. Организация машины на уровне ассемблера [обязательный]

Минимальное время, отводимое на раздел: 9 часов

Темы:

Принципы фон Неймана
Управляющее устройство: выборка инструкций, декодирование и выполнение
Набор инструкций и виды инструкций (манипуляция данными, управление, ввод-вывод)
Программирование на уровне ассемблера и машинном уровне
Форматы инструкций
Режимы адресации
Механизм вызовов и возвратов из процедур
Ввод-вывод и прерывания

Задачи обучения:

1. Объяснить организацию классической машины фон Неймана и ее основных функциональных блоков.
2. Объяснить, как происходит выполнение инструкции в машине фон Неймана.
3. Объяснить, как инструкции представляются на машинном уровне и на уровне ассемблера.
4. Объяснить различные форматы инструкций, в частности, форматы с фиксированной длиной и переменной длиной.
5. Написать небольшие фрагменты ассемблерной программы.
6. Продемонстрировать реализацию фундаментальных высокоуровневых конструкций программирования на машинном уровне.
7. Объяснить механизм поддержки вызовов процедур на машинном уровне.
8. Объяснить базовые понятия прерываний и операций ввода-вывода.

AR4. Устройство памяти компьютера [обязательный]

Минимальное время, отводимое на раздел: 5 часов

Темы:

Системы и технологии хранения данных
Кодирование данных, сжатие данных и целостность
Иерархия памяти
Организация основной памяти и операции с ней
Задержка, время цикла, пропускная способность и чередование
Кэш-память (отображение адресов, размер блока, стратегия замены и сохранения)
Виртуальная память (таблица страниц, буфер быстрого преобразования адреса)
Обработка ошибок и надежность

Задачи обучения:

1. Определить основные виды памяти.
2. Объяснить эффект задержки памяти во время выполнения.
3. Объяснить использование иерархии памяти для преодоления эффекта задержки.
4. Объяснить основы управления памятью.
5. Объяснить роль кэша и виртуальной памяти.
6. Объяснить принципы работы системы, использующей виртуальную память.

AR5. Взаимодействие и коммуникации [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Основы ввода-вывода: "рукопожатие", буферизация, программируемый ввод-вывод, ввод-вывод с прерываниями
Структуры прерываний: вектор прерываний, прерывания с приоритетами, подтверждение прерывания
Внешняя память, физическая организация и устройства
Шины: протоколы шины, голосование, прямой доступ к памяти
Введение в сетевые технологии
Поддержка мультимедиа
Архитектуры RAID

Задачи обучения:

1. Объяснить использование прерываний для реализации управления вводом-выводом и передачей данных.
2. Определить различные типы шин в компьютерных системах.
3. Объяснить доступ к данным с магнитного диска.
4. Сравнить распространенные сетевые конфигурации.
5. Определить интерфейсы, необходимые для поддержки мультимедиа.
6. Объяснить преимущества и ограничения архитектур RAID.

AR6. Функциональная организация [обязательный]

Минимальное время, отводимое на раздел: 7 часов

Темы:

Реализация простых информационных каналов
Устройство управления: "жесткая" реализация и микропрограммная реализация
Конвейерная обработка команд

Введение в параллелизм на уровне инструкций (ILP)

Задачи обучения:
1. Сравнить альтернативные реализации информационных каналов.
2. Обсудить понятие контрольной точки и генерацию сигналов управления с использованием "жесткой" и микропрограммной реализации.
3. Объяснить основы параллелизма на уровне инструкций с использованием конвейера и основные проблемы, которые могут возникнуть при этом.

AR7. Многопроцессорные и альтернативные архитектуры [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Введение в SIMD, MIMD, VLIW, EPIC
Систолическая архитектура
Сети с внутрисхемной коммутацией (гиперкуб, сеть с "перетасовкой", ячеистая архитектура, перекрестное соединение)
Системы с общей памятью
Когерентность кэшей
Модели памяти и модели целостности

Задачи обучения:
1. Обсудить понятие параллельной обработки в моделях, отличающихся от модели фон Неймана.
2. Описать альтернативные архитектуры, такие как SIMD, MIMD, и VLIW.
3. Объяснить понятие сетей коммутации и охарактеризовать различные подходы.
4. Обсудить проблемы управления памятью, возникающие в многопроцессорных системах и описать пути решения этих проблем.

AR8. Улучшение производительности [факультативный]

Темы:

Суперскалярные архитектуры
Предсказание переходов
Предварительная выборка
Спекулятивное выполнение
Многопоточность
Масштабируемость

Задачи обучения:
1. Описать суперскалярные архитектуры и их преимущества.
2. Объяснить понятие предсказания переходов и его значение.
3. Охарактеризовать стоимости и преимущества предварительной выборки.
4. Описать спекулятивное выполнение и условия, при которых оно оправдано.
5. Обсудить преимущества в производительности, которые дает многопоточность и факторы, препятствующие извлечению максимальной выгоды из многопоточности.
6. Описать связь масштабируемости и производительности.

AR9. Архитектура сетевых и распределенных систем [факультативный]

Темы:

Введение в локальные (LAN) и глобальные (WAN) сети
Многоуровневая модель сетевых протоколов, ISO/OSI, IEEE 802
Влияние архитектурных вопросов на распределенные алгоритмы
Сетевые вычисления
Распределенные мультимедиа-системы

Задачи обучения:
1. Объяснить основные компоненты сетевых систем и различия между локальными (LAN) и глобальными (WAN) сетями.
2. Обсудить архитектурные вопросы, связанные с проектированием многоуровневых сетевых протоколов.
3. Объяснить архитектурные отличия сетевых и распределенных систем.
4. Обсудить архитектурные вопросы, связанные с сетевыми вычислениями и распределенными мультимедиа-технологиями.

Операционные системы (OS)

OS1. Обзор операционных систем [обязательный]
OS2. Основы операционных систем [обязательный]
OS3. Параллелизм [обязательный]
OS4. Планирование и диспетчеризация [обязательный]
OS5. Управление памятью [обязательный]
OS6. Управление устройствами [факультативный]
OS7. Безопасность и защита данных [факультативный]
OS8. Файловые системы [факультативный]
OS9. Встроенные системы и системы реального времени [факультативный]
OS10. Отказоустойчивость [факультативный]
OS11. Оценка производительности системы [факультативный]
OS12. Языки сценариев [факультативный]

Операционная система (operating system) предоставляет программистам удобную абстракцию аппаратного обеспечения компьютера, а также управляет разделением ресурсов между пользователями. Темы данного раздела затрагивают вопросы, влияющие на проектирование современных операционных систем. Учебные курсы в данной области обычно включают практические занятия, позволяющие студентам экспериментировать с операционными системами.

В течение многих лет операционные системы и их абстракции становились все более сложными по сравнению с обычными прикладными программами. Прежде чем перейти к изучению реализацию внутренних алгоритмов и структур данных, необходимо убедиться, что студенты осознают, насколько широко в современном мире используются операционные системы. Поэтому данные темы связаны как с использованием операционных систем, так и с их проектированием и реализацией. Многие идеи, возникшие в процессе разработки операционных систем, нашли приложения в других областях информатики, например, в параллельном программировании. Изучение внутренней структуры операционных систем отражается на таких областях, как программирование с повышенными требованиями к надежности, проектирование и реализация алгоритмов, разработка современных устройств, создание виртуальных сред, кэширование документов в Internet, создание безопасных и защищенных систем, управление сетями и многих других.

OS1. Обзор операционных систем [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Роль и задачи операционных систем
История развития операционных систем
Функционирование типичной операционной системы
Механизмы поддержки модели "клиент-сервер", мобильных устройств
Вопросы проектирования (эффективность, надежность, гибкость, переносимость, безопасность, совместимость)
Влияние безопасности, сетевых технологий, мультимедиа, графических оконных интерфейсов

Задачи обучения:

1. Объяснить цели и функции современных операционных систем.
2. Описать, как операционные системы эволюционировали от примитивных пакетных систем к сложным многопользовательским операционным системам.
3. Проанализировать компромиссные решения, которые приходится принимать при проектировании операционных систем.
4. Описать функции современной операционной системы с точки зрения удобства работы, эффективности и масштабируемости.
5. Обсудить сетевые, клиент-серверные, распределенные операционные системы и их отличия от однопользовательских операционных систем.
6. Указать потенциальные проблемы с безопасностью операционных систем и возможные пути их решения.
7. Объяснить, как программное обеспечение с открытыми исходными текстами и широкое распространение Internet влияют на проектирование операционных систем.

OS2. Основы операционных систем [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Методы декомпозиции системы (монолитная, многоуровневая, модульная, микроядерная модели)
Абстракции, процессы и ресурсы
Понятие программного интерфейса приложения (API)
Потребности приложений и эволюция программного и аппаратного обеспечения
Организация устройств
Прерывания: методы и реализации
Понятие состояния системы и пользователя, переход в режим ядра

Задачи обучения:

1. Объяснить понятие логического уровня.
2. Объяснить преимущества организации системы в виде иерархии уровней абстракции.
3. Объяснить необходимость API и промежуточного ПО (middleware).
4. Объяснить, как вычислительные ресурсы используются прикладным ПО и управляются системным ПО.
5. Сопоставить пользовательский режим и режим ядра операционной системы.
6. Обсудить преимущества и недостатки использования прерываний.
7. Сравнить различные способы декомпозиции операционной системы: объектно-ориентированный, модульный, микроядерный и многоуровневый.
8. Объяснить применение списка устройств и очереди ввода-вывода драйверов.

OS3. Параллелизм [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Состояния и диаграммы состояний
Структуры (таблица готовности, блоки управления процессом…)
Диспетчеризация и переключение контекстов
Роль прерываний
Параллельное исполнение: преимущества и недостатки
Проблемы "взаимного исключения" и некоторые решения
Тупики: причины, условия, способы предотвращения
Модели и механизмы (семафоры, мониторы, условные переменные, рандеву)
Проблема "читатель/писатель" и синхронизация
Вопросы, связанные с многопроцессорностью (спин-блокировка, реентерабельность)

Задачи обучения:

1. Объяснить необходимость параллелизма внутри среды операционной системы.
2. Продемонстрировать потенциальные проблемы времени исполнения, возникающие из-за наличия множества параллельных операций в отдельных задачах.
3. Описать различные механизмы, которые могут использоваться на уровне операционной системы для реализации параллелизма, и достоинства каждого из них.
4. Объяснить различные состояния, через которые проходит задача, и структуры данных, необходимые для поддержки многозадачности.
5. Описать различные подходы к решению проблемы взаимного исключения в операционных системах.
6. Объяснить причины использования прерывания, диспетчеризации и переключения контекстов в операционной системе.
7. Изобразить состояния и диаграммы переходов для простых примеров.
8. Обсудить применение структур данных, таких как стеки и очереди, в управлении параллелизмом.
9. Описать условия возникновения тупиков.

OS4. Планирование и диспетчеризация [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Вытесняющее и невытесняющее планирование
Планировщики и политики
Процессы и потоки
Предельные сроки выполнения операции и другие вопросы реального времени

Задачи обучения:

1. Сравнить различные алгоритмы, использующиеся для вытесняющего и невытесняющего планирования задач в операционных системах, такие как использование приоритетов, сравнение производительности и "справедливые" схемы.
2. Описать отношения между алгоритмами диспетчеризации и различными предметными областями.
3. Обсудить типы планирования: краткосрочное, среднесрочное, долгосрочное и ввод-вывод.
4. Описать различия между процессами и потоками.
5. Сравнить статические и динамические подходы к планированию.
6. Обсудить необходимость приоритетного обслуживания и планирования по сроку завершения.
7. Указать примеры использование логики алгоритмов планирования в таких областях как дисковый ввод-вывод, сетевое планирование, планирование проектов и других задачах, не имеющих непосредственного отношения к компьютерам.

OS5. Управление памятью [обязательный]

Минимальное время, отводимое на раздел: 5 часов

Темы:

Обзор физической памяти и аппаратного обеспечения, предназначенного для управления памятью
Оверлеи, подкачка и разделы
Страничная и сегментная организации памяти
Стратегии вытеснения страниц
Рабочее множество и пробуксовка
Кэширование

Задачи обучения:

1. Описать иерархическое устройство памяти и компромисс между стоимостью и производительностью.
2. Объяснить понятие виртуальной памяти и ее реализацию на аппаратном и программном уровне.
3. Описать такие аспекты виртуальной памяти, как кэширование, страничную и сегментную организацию.
4. Оценить компромиссные решения в терминах размера памяти (основной памяти, кэш-памяти, вспомогательной памяти) и скорости процессора.
5. Описать различные методы выделения памяти задачам, отметить достоинства каждого из них.
6. Описать причину и способ использования кэш-памяти.
7. Сравнить страничную и сегментную организацию памяти.
8. Обсудить понятие пробуксовки, причины ее возникновения и методы обнаружения пробуксовки и борьбы с ней.
9. Проанализировать различные способы экономии памяти: оверлеи, подкачку и стратегии вытеснения.

OS6. Управление устройствами [факультативный]

Темы:

Характеристики последовательных и параллельных устройств
Абстрагирование от различий между устройствами
Стратегии буферизации
Прямой доступ к памяти
Восстановление после сбоев

Задачи обучения:

1. Объяснить ключевые отличия между последовательными и параллельными устройствами и причины выбора между ними.
2. Описать отношение между физическим аппаратным обеспечением и виртуальными устройствами, обеспечиваемыми операционной системой.
3. Описать буферизацию и стратегии ее реализации.
4. Привести различия между механизмами, используемыми для взаимодействия с устройствами компьютера (включая портативные устройства, сетевое оборудование, мультимедиа) и объяснить влияние этих различий на проектирование операционной системы.
5. Описать преимущества и недостатки прямого доступа к памяти и обсудить условия, в которых использование прямого доступа оправдано.
6. Определить требования к восстановлению после сбоев.
7. Реализовать простой драйвер для ряда устройств.

OS7. Безопасность и защита данных [факультативный]

Темы:

Обзор безопасности системы
Механизмы и политики разграничения прав доступа
Методы и устройства обеспечения безопасности
Защита, доступ и аутентификация
Модели защиты
Защита памяти
Шифрование данных
Управление восстановлением

Задачи обучения:

1. Объяснить необходимость безопасности и защиты данных, роль этических вопросов в использовании компьютера.
2. Описать свойства и ограничения операционной системы, используемые для обеспечения безопасности и защиты данных.
3. Сравнить современные методы реализации безопасности.
4. Сравнить сильные и слабые стороны двух или более популярных в настоящее время операционных систем с точки зрения безопасности.
5. Сравнить сильные и слабые стороны двух или более популярных в настоящее время операционных систем с точки зрения управления восстановлением.

OS8. Файловые системы [факультативный]

Темы:

Файлы: данные, метаданные, операции, организация, буферизация, файлы с последовательным и произвольным доступом
Директории: содержимое и структура
Файловые системы: деление на разделы, монтирование и демонтирование, виртуальные файловые системы
Стандартные методы реализации
Файлы, проецируемые в память
Специализированные файловые системы
Именование, поиск, доступ, резервное копирование

Задачи обучения:

1. Объяснить необходимость файловых систем.
2. Сравнить различные подходы к организации файлов, указать сильные и слабые стороны каждого подхода.
3. Объяснить, как развитие аппаратного обеспечения привело к изменениям приоритетов в вопросах проектирования файловых систем и управления ими.

OS9. Встроенные системы и системы реального времени [факультативный]

Темы:

Планирование процессов и задач
Требования к управлению памятью и дисковыми устройствами в среде реального времени
Сбои, риски и восстановление
Специфические проблемы систем реального времени

Задачи обучения:

1. Определить, какие черты делают систему системой реального времени.
2. Описать понятие задержки и характеристики задержки в системах реального времени.
3. Описать специфические проблемы, возникающие в системах реального времени и методы решения этих проблем.

OS10. Отказоустойчивость [факультативный]

Темы:

Фундаментальные понятия: надежность и доступность системы
Пространственная и временная избыточность
Методы реализации отказоустойчивости
Примеры надежных систем

Задачи обучения:

1. Объяснить важность и взаимоотношение понятий *отказоустойчивость, надежность* и *доступность.*
2. Обрисовать круг методов, применяющихся для обеспечения отказоустойчивости в операционных системах.
3. Объяснить, как операционная система может продолжить свою работу после сбоя.

OS11. Оценка производительности системы [факультативный]

Темы:

Почему требуется оценивать производительность системы
Какие параметры системы необходимо оценивать
Стратегии кэширования, подкачки, планирования, управления памятью, безопасности и другие
Модели оценки: детерминированная, аналитическая, имитационная и зависящая от реализации
Методы сбора данных для оценки (механизмы профилирования и трассировки)

Задачи обучения:

1. Описать метрики производительности, используемые для оценки производительности системы.
2. Объяснить основные модели оценки производительности системы.

OS12. Языки сценариев [факультативный]

Темы:

Языки сценариев (scripting languages) и их роль
Основные системные команды
Написание сценариев, передача параметров
Выполнение сценария
Влияние языков сценариев на программирование

Задачи обучения:

1. Описать стандартный набор команд, предоставляемый операционной системой.
2. Продемонстрировать типичные функции языка сценариев и их использование в программировании.
3. Описать механизмы реализации языков сценариев и роль этих языков в реализации и интеграции системы.
4. Написать небольшой сценарий, иллюстрирующий передачу параметров.

Распределенные вычисления (NC)

NC1. Введение в распределенные вычисления [обязательный]
NC2. Сети и телекоммуникации [обязательный]
NC3. Сетевая безопасность [обязательный]
NC4. Web как пример архитектуры "клиент-сервер" [обязательный]
NC5. Разработка Web-приложений [факультативный]
NC6. Управление сетями [факультативный]
NC7. Сжатие и распаковка данных [факультативный]
NC8. Технологии мультимедиа [факультативный]
NC9. Беспроводные и мобильные компьютеры [факультативный]

Последние достижения в области сетей и телекоммуникаций, особенно основанные на TCP/IP, увеличили значение сетевых технологий в компьютерной дисциплине. Распределенные вычисления (net-centric computing) объединяют набор дисциплин, включающий в себя: понятия и протоколы компьютерных коммуникаций, мультимедиа-системы, стандарты и технологии Web, сетевую безопасность, беспроводные и мобильные компьютеры и распределенные системы.

Владение сетевыми технологиями включает в себя как теоретические знания, так и практические навыки. Настоятельно рекомендуется, чтобы обучение включало в себя получение практического опыта и его анализ, поскольку это укрепляет у учащихся понимание концепций данной предметной области и их приложений к задачам реального мира. Лабораторный опыт должен включать в себя сбор и синтез данных, моделирование, анализ протоколов на уровне исходных текстов, мониторинг сетевых пакетов, написание программ и оценку возможных вариантов проектирования. Все эти важные понятия лучше усваиваются в процессе выполнения лабораторных работ.

NC1. Введение в распределенные вычисления [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Основы и история компьютерных сетей и Internet
Сетевые архитектуры
Специализации распределенных вычислений
– Сети и протоколы
– Сетевые мультимедиа-системы
– Распределенные вычисления
– Беспроводные и мобильные компьютеры

Задачи обучения:

1. Обсудить эволюцию первых сетей и Internet.
2. Научиться эффективно использовать ряд распространенных сетевых приложений, включая электронную почту, telnet, группы новостей, web-браузеры, web-курсы и системы мгновенной передачи сообщений.
3. Описать иерархическую многоуровневую структуру сетевых архитектур.
4. Описать новые технологии распределенных вычислений, оценить, по состоянию на сегодняшний день, их возможности, ограничения и потенциал.

NC2. Сети и телекоммуникации [обязательный]

Минимальное время, отводимое на раздел: 7 часов

Темы:

Сетевые стандарты и учреждения, занимающиеся стандартизацией
Семиуровневая модель сетевых протоколов ISO и ее отражение в TCP/IP
Коммутация каналов и пакетная коммутация
Потоки и дейтаграммы
Понятия физического уровня сетевых протоколов (теоретические основы, средства передачи данных, стандарты)
Понятия канального уровня (кадрирование, контроль за ошибками, управление потоками, протоколы)
Объединение сетей и маршрутизация (алгоритмы маршрутизации, обеспечение межсетевого обмена, контроль перегрузки)
Службы транспортного уровня (установка соединения, проблемы производительности)

Задачи обучения:

1. Обсудить наиболее важные сетевые стандарты в их историческом контексте.
2. Описать функции первых четырех уровней модели ISO.
3. Обсудить различия между коммутацией каналов и пакетной коммутацией, достоинства и недостатки обоих методов.
4. Объяснить, как сеть может обнаруживать и исправлять ошибки передачи данных.
5. Описать, как пакеты маршрутизируются в сетях Internet.
6. Установить простую сеть с двумя клиентами и одним сервером, использующую стандартные средства конфигурации, такие как DHCP.

NC3. Сетевая безопасность [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Основы криптографии
Алгоритмы с секретным ключом
Алгоритмы с открытым ключом
Протоколы аутентификации
Цифровые подписи
Примеры

Задачи обучения:

1. Обсудить основные принципы криптографии с открытым ключом.
2. Описать, как работают криптографические алгоритмы с открытым ключом.
3. Описать различия между алгоритмами с секретным и открытым ключом.
4. Описать классические протоколы аутентификации.

5. Сгенерировать и распространить пару ключей, использовать PGP-пакет для посылки зашифрованного сообщения по электронной почте.
6. Описать возможности и ограничения широко распространенных криптографических методов.

NC4. Web как пример архитектуры "клиент-сервер" [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Web-технологии:
– Программное обеспечение Web-серверов
– Программы CGI (Common gateway interface)
– Сценарии, выполняющиеся на клиентской стороне
– Понятие апплета
Характеристики web-серверов
– Поддержка механизма прав доступа
– Управление файлами
– Возможности распространенных серверных архитектур
Роль клиентских компьютеров
Свойства отношений "клиент-сервер"
Web-протоколы
Программные средства поддержки разработки Web-сайтов и управления web
Разработка Web-серверов
Опубликование информации в Web

Задачи обучения:

1. Объяснить различные роли и ответственности клиентов и серверов для различных приложений.
2. Выбрать набор программ, обеспечивающих эффективность реализации различных возможностей клиент-серверной архитектуры.
3. Разработать простое интерактивное web-приложение (например, форму, позволяющую ввести информацию на стороне клиента и сохранить ее на сервере в виде файла).

NC5. Разработка Web-приложений [факультативный]

Темы:

Протоколы уровня приложений
Основы web-инженерии
Web-сайты, основанные на базах данных
Удаленный вызов процедур (RPC)
"Легкие" распределенные объекты
Роль промежуточного ПО
Инструментальные средства поддержки
Проблемы безопасности в распределенных объектных средах
Web-приложения масштаба предприятия

Задачи обучения:

1. Показать, как интерактивные клиент-серверные web-приложения среднего размера могут быть написаны с использованием различных типов Web-технологий.
2. Продемонстрировать, как реализовать web-сайт, основанный на базе данных, с объяснением технологий, используемых при реализации каждого уровня архитектуры и соответствующие компромиссные решения в отношении производительности.
3. Реализовать два варианта распределенной системы с использованием двух распределенных объектных сред и сравнить их с точки зрения производительности и безопасности.
4. Обсудить проблемы безопасности и методы ее обеспечения в web-приложениях масштаба предприятия.

NC6. Управление сетями [факультативный]

Темы:

Обзор проблем сетевого управления
Использование паролей и механизмов контроля доступа
Доменные имена и служба имен
Проблемы поставщиков услуг Internet (ISPs)
Вопросы безопасности и брандмауэры
Вопросы качества обслуживания: производительность, восстановление после сбоев

Задачи обучения:

1. Объяснить проблемы управления сетями, возникающие из-за угроз безопасности, включая вирусы, "червей", Троянских коней и атак, направленных на инициирование отказов в обслуживании.
2. Указать сильные и слабые стороны различных подходов к обеспечению безопасности.
3. Разработать стратегию обеспечения требуемого уровня безопасности в специализированной системе.
4. Реализовать сетевой брандмауэр.

NC7. Сжатие и распаковка данных [факультативный]

Темы:

Аналоговое и цифровое представление данных
Упаковка и распаковка данных
Сжатие с потерей информации и без потерь
Сжатие данных: кодирование Хаффмана и алгоритм Лемпель-Зива
Сжатие и распаковка аудиоданных
Сжатие и распаковка изображений
Сжатие и распаковка видеоданных
Проблемы производительности: синхронизация, коэффициент сжатия, пригодность для использования в условиях реального времени

Задачи обучения:

1. Описать основные характеристики выборки и квантования для цифрового представления.
2. Объяснить, какие методы сжатия текстовой, аудио, видеоинформации и изображений наиболее пригодны для данного приложения в данных условиях и почему.
3. Описать асимметричное свойство алгоритмов сжатия и распаковки данных.
4. Проиллюстрировать понятие кодирования длин серий.
5. Показать, как программы, подобные UNIX-утилите compress, использующей кодирование Хаффмана и алгоритм Лемпель-Зива, сжимают типичный текстовый файл.

NC8. Технологии мультимедиа [факультативный]

Темы:

Звук и аудио, изображения и графика, анимация и видео
Стандарты мультимедиа (аудио, музыка, графика, изображения, телефония, видео, телевидение)
Планирование загрузки и проблемы производительности
Устройства ввода и вывода (сканеры, цифровые камеры, сенсорные экраны и устройства, управляемые голосом)
MIDI-клавиатуры, синтезаторы
Стандарты хранения информации (магнитооптические накопители, CD-ROM, DVD)
Мультимедиа серверы и файловые системы
Инструментальные средства поддержки мультимедийных разработок

Задачи обучения:

1. Для каждого из нескольких медиа- или мультимедиа стандартов описать требования этого стандарта на языке, понятном неспециалисту, и объяснить, как ограничения этого стандарта могут быть восприняты человеком.
2. Оценить потенциал компьютерной системы с точки зрения одного из мультимедийных приложений, включая оценку требований мультимедийной системы к сетевым технологиям.
3. Описать характеристики компьютерной системы (включая инструментальные средства поддержки и соответствующие стандарты), требуемой для выполнения данного мультимедийного приложения.
4. Реализовать небольшое мультимедийное приложение.

NC9. Беспроводные и мобильные компьютеры [факультативный]

Темы:

История, эволюция и совместимость беспроводных стандартов
Специфические проблемы беспроводных и мобильных компьютеров
Беспроводные локальные сети и сети, основанные на спутниковой связи
Беспроводные линии связи
Протоколы Internet для мобильных устройств
Адаптация к условиям мобильных устройств
Расширения клиент-серверной модели для адаптации ее к мобильным устройствам
Мобильный доступ к данным: распространение данных с сервера и управление кэшем на клиенте

Установка программ на мобильные и беспроводные компьютеры
Роль промежуточного ПО и инструментальных средств поддержки
Проблемы производительности
Новые технологии

Задачи обучения:

1. Описать основные характеристики мобильного IP и отличия от обычного IP с точки зрения мобильности и управления местоположением, а также производительности.
2. Проиллюстрировать с помощью домашних и удаленных агентов механизм маршрутизации электронной почты и других видов трафика с использованием мобильного IP.
3. Реализовать небольшое приложение, использующее мобильные и беспроводные коммуникации.
4. Описать области применения мобильных компьютеров в настоящее время и в перспективе, оценить возможности, ограничения и потенциал каждой из этих областей.

Языки программирования (PL)

PL1. Обзор языков программирования [обязательный]
PL2. Виртуальные машины [обязательный]
PL3. Введение в трансляцию [обязательный]
PL4. Переменные и типы данных [обязательный]
PL5. Механизмы абстракции [обязательный]
PL6. Объектно-ориентированное программирование [обязательный]
PL7. Функциональное программирование [факультативный]
PL8. Системы трансляции [факультативный]
PL9. Системы типов [факультативный]
PL10. Семантика языков программирования [факультативный]
PL11. Разработка языков программирования [факультативный]

Языки программирования (programming languages) являются основным средством общения программиста и компьютера. Программисты должны не просто уметь написать программу на каком-либо одном языке, они должны понимать различные стили программирования, присущие разным языкам. На протяжении своей профессиональной карьеры программисты будет работать с множеством различных языков и стилей одновременно. Понимание разнообразия языков программирования и различных парадигм значительно облегчает быстрое освоение новых языков. Для понимания прагматических аспектов языков программирования требуются также базовые знания теории трансляции языков программирования и механизмов работы среды времени выполнения, например, распределения памяти.

PL1. Обзор языков программирования [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

История языков программирования
Краткий обзор парадигм программирования
– Процедурные языки
– Объектно-ориентированные языки
– Функциональные языки
– Декларативные, неалгоритмические языки
– Языки сценариев
Влияние масштаба решаемых задач на методологию программирования масштаба разработок

Задачи обучения:

1. Описать эволюцию языков программирования, показать, как в результате исторического развития появились современные парадигмы программирования.
2. Указать, по крайней мере, одну отличительную особенность каждой из парадигм программирования, встречающихся в этом разделе.
3. Оценить преимущества и недостатки выбора каждой из парадигм, принимая во внимание такие вопросы, как используемая память, время (как компьютерное, так и программиста), безопасность и выразительная сила.
4. Описать отличия между "программированием-в-малом" и "программированием-в-большом".

PL2. Виртуальные машины [обязательный]

Минимальное время, отводимое на раздел: 1 час

Темы:

Понятие виртуальной машины
Иерархия виртуальных машин
Промежуточные языки
Проблемы безопасности, связанные с выполнением кода на сторонней машине

Задачи обучения:

1. Объяснить важность и силу абстракции в контексте виртуальных машин.
2. Объяснить преимущества использования промежуточных языков в процессе компиляции.
3. Оценить компромисс между переносимостью и производительностью.
4. Объяснить, как исполняемые программы могут нарушить безопасность компьютерной системы путем доступа к дисковым файлам и памяти.

PL3. Введение в трансляцию [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Сравнение интерпретаторов и компиляторов
Стадии трансляции (лексический анализ, синтаксический анализ, генерация кода, оптимизация)
Машинно-зависимые и машинно-независимые аспекты трансляции

Задачи обучения:

1. Сравнить модели трансляции, ориентированные на компиляцию и интерпретацию, описать преимущества каждой из них.
2. Описать стадии трансляции программы от исходного кода до исполняемого модуля и файлы, генерируемые различными проходами.
3. Описать отличия между машинно-зависимой и машинно-независимой трансляцией и проявления этих различий в процессе трансляции.

PL4. Переменные и типы данных [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Концепция типа данных как множества значений и операций над ними
Свойства объявлений (связывание, область видимости, блоки и время жизни)
Обзор проверки типов
Сборка мусора

Задачи обучения:

1. Объяснить значение различных моделей описаний данных, особенно в применении к "программированию-в-большом".
2. Описать различные свойства переменной, включая адрес переменной, значение, область видимости, "живучесть" и размер, а также объяснить их смысл.
3. Обсудить понятие несовместимости типов.
4. Продемонстрировать различные формы использования связывания, областей видимости и управления временем жизни.
5. Объяснить важность типов и проверки типов для абстракции и безопасности.
6. Описать различные способы управления временем жизни переменных (подсчет ссылок и сборка мусора).

PL5. Механизмы абстракции [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Процедуры, функции и итераторы как механизмы абстракции
Механизмы параметризации (ссылки и значения)
Активационные записи и управление памятью
Параметры типов и параметризованные типы

Модули в языках программирования

Задачи обучения:
1. Объяснить, как механизмы абстракции поддерживают создание повторно используемых компонент программного обеспечения.
2. Показать отличия между передачей параметров по значению и по ссылке.
3. Объяснить важность абстракций, особенно в применении к "программированию-в-большом".
4. Описать, как используются активационные записи для управления программными модулями и их данными.

PL6. Объектно-ориентированное программирование [обязательный]

Минимальное время, отводимое на раздел: 10 часов

Темы:

Объектно-ориентированное проектирование
Инкапсуляция и сокрытие информации
Разделение поведения и реализации
Классы и подклассы
Наследование (переопределение, динамическое связывание)
Полиморфизм (полиморфизм подтипов и наследование)
Иерархии классов
Классы коллекций и протоколы итерации
Внутреннее представление объектов и таблиц методов

Задачи обучения:
1. Описать философию объектно-ориентированного проектирования и понятия инкапсуляции, абстракции, наследования и полиморфизма.
2. Спроектировать, реализовать, протестировать и отладить небольшие программы на объектно-ориентированном языке.
3. Объяснить, как механизм классов поддерживает инкапсуляцию и скрытие информации.
4. Спроектировать, реализовать и протестировать реализацию отношения "is-a" между объектами с использованием иерархии классов и наследования.
5. Сравнить понятия перегрузки и переопределения методов в объектно-ориентированном языке.
6. Объяснить отношение между статической структурой класса и динамической структурой экземпляров класса.
7. Описать, как итераторы получают доступ к элементам контейнера.

PL7. Функциональное программирование [факультативный]

Темы:

Обзор и мотивация функциональных языков
Рекурсия: списки, натуральные числа, деревья и другие рекурсивно определяемые структуры данных
Прагматика (отладка методом "разделяй и властвуй", живучесть структур данных)
Амортизационная эффективность для функциональных структур данных
Замыкания и использование функций в качестве данных (бесконечные множества, потоки)

Задачи обучения:
1. Описать сильные и слабые стороны функциональной парадигмы программирования.
2. Спроектировать, написать, протестировать и отладить несколько программ с использованием функциональной парадигмы.
3. Пояснить использование функций в качестве данных, включая понятие замыкания.

PL8. Системы трансляции [факультативный]

Темы:

Применение регулярных выражений в лексических анализаторах
Синтаксический анализ (конкретный и абстрактный синтез, абстрактные синтаксические деревья)
Применение контекстно-свободных грамматик в табличном синтаксическом анализе и методе рекурсивного спуска
Управление таблицей символов
Генерация кода путем обхода дерева
Архитектурно-зависимые оптимизации: выбор инструкций и распределение регистров
Методы оптимизации

Использование инструментальных средств для поддержки процесса трансляции и преимущества такого подхода
Библиотеки программ и раздельная компиляция
Создание синтаксически управляемых инструментальных средств

Задачи обучения:
1. Описать этапы трансляции и алгоритмы, применяемые в трансляторах.
2. Описать формальные модели, лежащие в основе методов трансляции (конечные автоматы, магазинные автоматы) и их связь с определением языков с помощью регулярных выражений и грамматик.
3. Обсудить эффективность оптимизации.
4. Описать влияние возможности раздельной компиляции и библиотек на процесс компиляции.

PL9. Системы типов [факультативный]

Темы:
Тип данных как набор значений и операций над ними
Типы данных
– Элементарные типы
– Производные и сопроизводные типы
– Алгебраические типы
– Рекурсивные типы
– Функциональные типы
– Параметризованные типы
Модели проверки типов
Семантические модели типов, определяемых пользователем
– Сокращенные типы
– Абстрактные типы данных
– Эквивалентность типов
Параметрический полиморфизм
Полиморфизм подтипов
Алгоритмы проверки типов

Задачи обучения:
1. Формализовать понятие типизации.
2. Описать элементарные типы данных.
3. Объяснить концепцию абстрактного типа данных.
4. Объяснить важность типизации для абстракции и безопасности.
5. Описать различия между статической и динамической типизацией.
6. Описать различия между явным объявлением типов и выведением типов.
7. Классифицировать языки с точки зрения типизации.

PL10. Семантика языков программирования [факультативный]

Темы:
Неформальная семантика
Обзор формальной семантики
Денотационная семантика
Аксиоматическая семантика
Операционная семантика

Задачи обучения:
1. Объяснить важность формальной семантики.
2. Описать отличия между формальной и неформальной семантикой.
3. Описать и оценить различные подходы к определению формальной семантики.

PL11. Разработка языков программирования [факультативный]

Темы:
Общие принципы проектирования языков
Задачи проектирования
Модели типизации
Модели структур данных
Модели структур управления
Механизмы абстракции

Задачи обучения:

1. Показать важность влияния различных моделей типизации на проектирование и использование языков программирования, а также на процесс трансляции.
2. Объяснить роль различных механизмов абстракции для создания средств, определенных пользователем.

Взаимодействие человека и машины (HC)

HC1. Основы взаимодействия человека и машины [обязательный]
HC2. Построение простого графического интерфейса [обязательный]
HC3. Оценка программного обеспечения, ориентированного на человека [факультативный]
HC4. Разработка программного обеспечения, ориентированного на человека [факультативный]
HC5. Проектирование графического интерфейса пользователя [факультативный]
HC6. Программирование графического интерфейса пользователя [факультативный]
HC7. Человеко-машинные аспекты мультимедиа-систем [факультативный]
HC8. Человеко-машинные аспекты сотрудничества и коммуникаций [факультативный]

Перечисленный в данном разделе набор тем служит введением в проблемы человеко-машинного взаимодействия (human-computer interaction, HCI) для студентов, специализирующихся в области информатики. Основной упор делается на понимании взаимодействия человека с интерактивными объектами, знании приемов разработки и оценки интерактивного программного обеспечения, ориентированного на человека, а также на общих знаниях вопросов проектирования человеко-машинных интерфейсов для многих видов программного обеспечения. Разделы HC1 (Основы взаимодействия человека и машины) и HC2 (Построение простого графического интерфейса) являются обязательными для всех учащихся и могут преподаваться в рамках вводных курсов. Остальные разделы, скорее всего, будут включены в один или два факультативных курса, предназначенных для студентов старших курсов.

HC1. Основы взаимодействия человека и машины [обязательный]

Минимальное время, отводимое на раздел: 6 часов

Темы:

Мотивация: почему следует беспокоиться о людях?
Аспекты человеко-машинного взаимодействия (инструментальные средства, гипермедиа в web, коммуникации)
Ориентированные на человека разработка и оценка программного обеспечения
Человеческие модели производительности: восприятие, движение и узнавание
Человеческие модели производительности: культура, коммуникации и организации
Приспособление к человеческому разнообразию
Принципы хорошего дизайна и хороших проектировщиков; инженерные компромиссы
Введение в тестирование на пригодность к использованию (usability testing)

Задачи обучения:

1. Обсудить причины разработки программного обеспечения, ориентированного на человека.
2. Описать основы науки психологического и социального взаимодействия.
3. Объяснить разницу между ролью гипотез и экспериментальных результатов с точки зрения их корреляции с результатом.
4. Создать концептуальный словарь для анализа взаимодействия человека и программ: приемлемость, концептуальная модель, обратная связь и так далее
5. Описать различия в интерпретации для данной иконки, символа, слова или цвета с точки зрения (a) двух различных культур и (b) культуры и одной из ее субкультур.
6. Показать пути успешного или неудачного проектирования компьютерной системы или приложения в терминах человеческого разнообразия.
7. Создать и провести простой тест на пригодность к использованию для существующего приложения.

HC2. Построение простого графического интерфейса [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Основы графических интерфейсов пользователя (GUI)
Инструментальные средства для разработки GUI

Задачи обучения:

1. Указать несколько ключевых принципов разработки эффективного графического интерфейса пользователя.
2. Использовать инструментальные средства для разработки GUI с целью создания простого приложения с графическим интерфейсом пользователя.
3. Показать эффект фундаментальных принципов проектирования на структуру графического интерфейса пользователя.
4. Провести простой тест на пригодность к использованию каждого элемента графического интерфейса и сравнить результаты.

HC3. Оценка программного обеспечения, ориентированного на пользователя [факультативный]

Темы:

Выявление задач оценивания
Оценка без пользователей: сквозные просмотры, KLM, руководящие документы и стандарты
Оценка с участием пользователей: тестирование на пригодность к использованию, интервью, обзор, эксперимент

Задачи обучения:

1. Обсудить критерии оценки: легкость обучения, время выполнения задачи, приемлемость.
2. Провести сквозной просмотр и анализ с помощью моделирования нажатий на клавиши (Keystroke Level Model , KLM).
3. Описать основные руководящие документы и стандарты.
4. Провести тест на пригодность к использованию, интервью и обзор.
5. Сравнить тест на пригодность к использованию с управляемым экспериментом.
6. Оценить существующую интерактивную систему с точки зрения критерия, ориентированного на человека, и теста на пригодность к использованию.

HC4. Разработка программного обеспечения, ориентированного на человека [факультативный]

Темы:

Подходы, характеристики и обзор процесса
Функциональность и пригодность к использованию: анализ задач, интервью, обзоры
Спецификация взаимодействия и представления
Методы и средства прототипирования
– "Бумажное" прототипирование
– Наследование и динамическое связывание
– Языки прототипирования и средства построения GUI

Задачи обучения:

1. Объяснить основные черты разработки, ориентированной на человека.
2. Сравнить разработку, ориентированную на человека, с традиционными моделями разработки программного обеспечения.
3. Описать три функциональных требования и три требования с точки зрения пригодности к использованию.
4. Специфицировать интерактивный объект с помощью сетей переходов, объектно-ориентированных средств или сценариев.
5. Обсудить достоинства и недостатки "бумажного" метода разработки и прототипирования.

HC5. Проектирование графического интерфейса пользователя [факультативный]

Темы:

Выбор стилей и методов взаимодействия с пользователем
Аспекты HCI для типовых экранных элементов
Аспекты HCI при проектировании экрана: размещение компонент, цвет, шрифты, обозначения
Обработка ошибок пользователя
Более сложные аспекты интерфейса: визуализация, представление, метафора
Способы взаимодействия: графика, звук и осязание
3D-интерфейс и виртуальная реальность

Задачи обучения:

1. Описать типичные стили взаимодействия.

2. Объяснить основные принципы хорошего дизайна для следующих элементов: экранные элементы; экранные формы; простой диалог с сообщением об ошибке; руководство пользователя.
3. Спроектировать, создать прототип и оценить простой двухмерный GUI, иллюстрирующий знание понятий, изученных в HC3 и HC4.
4. Обсудить трудности, возникающие при переходе с двухмерного на трехмерный интерфейс.

HC6. Программирование графического интерфейса пользователя [факультативный]

Темы:

Системы управления взаимодействием с пользователем (UIMS), независимость диалога и уровни анализа, модель Сихайма
Классы экранных элементов
Управление событиями и взаимодействием с пользователем
Управление геометрией
Средства создания GUI и UI-среды программирования
Кросс-платформенное проектирование

Задачи обучения:

1. Описать отличия между обязанностями UIMS и приложения.
2. Описать отличия между клиент-серверной моделью пользовательского интерфейса и моделью, основанной на ядре.
3. Сравнить парадигму программирования, управляемого событиями, с более традиционной процедурной парадигмой пользовательского интерфейса.
4. Описать агрегирование экранных элементов и управление геометрией, основанное на ограничениях.
5. Описать понятие обратного вызова и его роль в построителях GUI.
6. Указать, по меньшей мере, три различия, характерных для кросс-платформенных пользовательских интерфейсов.
7. Указать как можно больше общих черт пользовательских интерфейсов для различных платформ.

HC7. Человеко-машинные аспекты мультимедиа-систем [факультативный]

Темы:

Классификация и архитектуры информации: иерархии, гипермедиа
Извлечение информации и производительность действий человека
– Поиск в Web
– Использование языков запросов к базам данных
– Графика
– Звук
Проектирование HCI для информационных мультимедиа-систем

Распознавание речи и обработка естественного языка
Применение информации и мобильные компьютеры

Задачи обучения:

1. Обсудить, чем получение информации отличается от обработки транзакций.
2. Объяснить, как организация информации способствует ее эффективному получению.
3. Описать основные проблемы, которые возникают при использовании языков запросов к базам данных.
4. Описать современное состояние методов обработки естественного языка (в частности, в области распознавания речи).
5. Спроектировать, создать прототип и оценить простую информационную мультимедиа-систему, иллюстрирующую знание понятие, изученных в HC4, HC5 и HC7.

HC8. Человеко-машинные аспекты сотрудничества и коммуникаций [факультативный]

Темы:

Технологии совместного использования, предназначенные для выполнения специальных задач: подготовка документов, многопользовательские игры
Асинхронные средства коммуникаций внутри коллектива: электронная почта, электронные доски объявлений
Синхронные средства коммуникаций внутри коллектива: дискуссионные комнаты, конференции
Online-сообщества: MUDs/MOOs
Виртуальные персонажи и интеллектуальные агенты

Задачи обучения:

1. Сравнить вопросы человеко-машинного взаимодействия с точки зрения индивидуального пользователя и при групповом взаимодействии.
2. Обсудить проблемы социального характера, возникающие при использовании ПО коллективного использования.
3. Обсудить вопросы человеко-машинного взаимодействия в программном обеспечении, реализующем намерения человека.
4. Описать отличия между синхронными и асинхронными коммуникациями.
5. Спроектировать, создать прототип и оценить простую программу, предназначенную для коллективного использования и иллюстрирующую понятия, изученные в HC4, HC5 и HC8.
6. Принять участие в проекте, в котором взаимодействие между участниками осуществляется как непосредственно, так и с помощью специализированного программного обеспечения.
7. Описать общие черты и отличия между непосредственным взаимодействием и взаимодействием с помощью специализированных программ

Компьютерная графика и визуализация данных (GV)

GV1. Фундаментальные методы в графике [обязательный]
GV2. Графические системы [обязательный]
GV3. Графические коммуникации [факультативный]
GV4. Геометрическое моделирование [факультативный]
GV5. Основы рендеринга [факультативный]
GV6. Углубленное изучение рендеринга [факультативный]
GV7. Более сложные методы [факультативный]
GV8. Компьютерная анимация [факультативный]
GV9. Визуализация [факультативный]
GV10. Виртуальная реальность [факультативный]
GV11. Компьютерное зрение [факультативный]

Раздел, посвященный на компьютерной графике и визуализации, разделен на 4 части:

• *Компьютерная графика.* Компьютерная графика – это и искусство, и наука, посвященные использованию изображений, которые генерируются и представляются с помощью компьютеров. Это требует: (a) разработки моделей представления информации в виде, пригодном для создания и просмотра изображений, (b) разработки устройств и методов, с помощью которых люди могут взаимодействовать с моделью или представлением, (c) разработки методов рендеринга модели, и (d) разработки средств сохранения изображений. Задача компьютерной графики – вовлечь в процесс когнитивный процесс органы зрения человека.

• *Визуализация.* Задача визуализации – выявление и представление структур и отношений в наборах данных, относящихся к определенной научной области (информатике, медицине) или более абстрактных наборах данных. Важнейшей задачей представления должно быть улучшение восприятия информации. Хотя современные методы визуализации в основном используют зрительные способности человека, другие виды чувств, включая слух и осязание, также могут способствовать усвоению информации.

• *Виртуальная реальность.* Виртуальная реальность (VR) позволяет создать трехмерную среду, использующую компьютерную графику, и, возможно, другие органы чувств, для обеспечения лучшего взаимодействия между человеком и компьютерным миром.

• *Компьютерное зрение.* Задачей компьютерного зрения является воссоздание свойств и структуры трехмерного мира из одной и более двухмерных картинок. Понимание и применение компьютерного зрения основывается на ключевых понятиях в области информатики, но также сильно связано с дисциплинами физики, математики и психологии.

GV1. Фундаментальные методы в графике [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Иерархия графического программного обеспечения
Использование графических API
Простые цветовые модели (RGB, HSB, CMYK)
Однородные координаты
Аффинные преобразования (масштабирование, вращение, перенос)

Преобразования представления
Отсечение (clipping)

Задачи обучения:
1. Описать возможности различных уровней графического ПО и обсудить применимость каждого их них.
2. Создание изображений с использованием стандартных графических API.
3. Использование средств, обеспечиваемых стандартными API, для выражения основных преобразований: масштабирования, вращения, и переноса.
4. Реализовать простые процедуры, выполняющие операции преобразования и отсечения на простом двумерном изображении.
5. Обсудить трехмерную координатную систему и изменения, необходимые для расширения двумерных преобразований в трехмерные

GV2. Графические системы [обязательный]

Минимальное время, отводимое на раздел: 1 час

Темы:
Системы растровой и векторной графики
Устройства вывода видеоинформации
Физические и логические входные устройства
Проблемы, с которыми сталкиваются разработчики графических систем

Задачи обучения:
1. Описать применимость графической архитектуры для данного приложения.
2. Объяснить функции различных устройств ввода.
3. Сравнить методы растровой и векторной графики.
4. Научиться использовать современное аппаратное и программное обеспечение для создания и вывода графики.
5. Обсудить растущие возможности аппаратного и программного обеспечения для создания и вывода графики.

GV3. Графические коммуникации [факультативный]

Темы:
Психодинамика цвета и взаимодействия между цветами
Модификации цвета для людей с недостатками зрения
Культурное значение различных цветов
Использование эффективной псевдоцветовой палитры для изображений, рассчитанных на заданную аудиторию
Структурирование представлений для улучшения восприятия
Модификация изображений для эффективного представления в форме видеоинформации или тиражирования на бумаге
Использование легенды для представления информации о смысле цветов или других визуальных данных
Использование текста в изображениях для представления контекста и фоновой информации
Обратная связь с пользователем при графических операциях

Задачи обучения:
1. Объяснить значение использования цветов и псевдоцветов.
2. Продемонстрировать способность создавать эффективные видео- и печатные копии.
3. Указать примеры эффективных и неэффективных коммуникаций с использованием графики.
4. Привести грамотные примеры графических коммуникаций, правильно используя цвета, легенду, текст и/или видео.
5. Привести два примера с одним и тем же содержанием: первый для представления в виде бумажной копии, второй – для компьютерной презентации.
6. Обсудить различия в проектировании примеров, рассчитанных на представление в виде бумажной копии и в виде компьютерной презентации.

GV4. Геометрическое моделирование [факультативный]

Темы:
Полигональное представление трехмерных объектов
Параметрические полигональные кривые и поверхности
Представление с помощью конструктивной блочной геометрии (CSG)
Неявное представление кривых и поверхностей

Методы пространственного подразделения
Процедурные модели
Деформируемые модели
Поверхности подразделения
Моделирование с разным уровнем детализации
Реконструкция

Задачи обучения:
1. Создание простых многогранных моделей путем разбиения плоскостей.
2. Создание CSG-моделей из примитивов, таких как кубы и поверхности второго порядка (квадрики).
3. Представление объекта в виде многогранников по неявно заданной поверхности.
4. Создание фрактальной модели или местности с помощью процедурного метода.
5. Создание модели объекта по данным, полученным с помощью лазерного сканера.

GV5. Основы рендеринга [факультативный]

Темы:

Алгоритмы генерации прямых (Брезенхэм)
Генерация шрифтов: контурные и растровые шрифты
Источники света и свойства материалов
Рассеянное, диффузное и зеркальное отражение
Модель отражения Фонга
Растеризация полигональной поверхности; плоская заливка, заливка по методам Гуро и Фонга
Текстурирование, рельефная текстура, текстура среды
Введение в трассировку лучей
Синтез изображений, методы семплирования и сглаживание

Задачи обучения:
1. Объяснить действия алгоритма Брезенхэма по отображению прямой на точечном экране.
2. Объяснить суть и приложения каждого из перечисленных выше методов.
3. Продемонстрировать каждый из методов на примере путем создания изображения с помощью стандартного API.
4. Описать процесс создания изображения.

GV6. Углубленное изучение рендеринга [факультативный]

Темы:

Уравнения переноса
Алгоритмы трассировки лучей
Трассировка фотонов
Учет эффекта освещения отраженным светом (radiosity) для вычисления общей освещенности, зависимость от формы
Эффективные подходы к расчету глобальной освещенности
Методы Монте-Карло для расчета глобальной освещенности
Рендеринг на основе изображения, панорамный вид
Рендеринг сложных явлений природы
Нефотореалистичный рендеринг

Задачи обучения:
1. Подробно описать несколько уравнений переноса с учетом всевозможных деталей.
2. Описать эффективные алгоритмы для вычисления освещенным отраженным светом и компромиссы между точностью и производительностью.
3. Описать влияние различных схем представления объекта на конечное изображение.
4. Объяснить методы рендеринга на основе изображений, текстур освещенности и связанные с ними вопросы.

GV7. Более сложные методы [факультативный]

Темы:

Квантование цвета
Сканирующее преобразование двумерного примитива
Разбиение кривых поверхностей
Методы удаления скрытых поверхностей
Z-буфер и кадровый буфер, цветовые каналы (альфа-канал)

Более сложные методы геометрического моделирования

Задачи обучения:
1. Описать методы, упомянутые в этом разделе.
2. Объяснить, как определить графические методы, использованные для создания данного изображения.
3. Реализовать любой из упомянутых графических методов с использованием примитивов графической системы на уровне отдельной точки.
4. Использовать программное обеспечение для анимации для создания простых органических форм с использованием графических примитивов и скелетов.

GV8. Компьютерная анимация [факультативный]

Темы:

Анимация по ключевым кадрам
Анимация камеры
Система сценариев
Анимация сочлененных структур: обратная кинематика
Захват движения (motion capture)
Процедурная анимация
Деформация

Задачи обучения:
1. Объяснить метод интерполяции с помощью сплайнов для расчета промежуточных позиций и ориентации.
2. Сравнить различные технологии захвата движения.
3. Создать простую анимационную картину (например, фейерверк) с помощью программного обеспечения для анимации, применяя систему частиц.
4. Использовать методы деформации свободной формы для создания различных деформаций.

GV9. Визуализация [факультативный]

Темы:

Визуализация векторных полей, тензоров и потоковых данных
Визуализация скалярных полей или полей высот
Простая визуализация объемных данных: трассировка лучей, функции преобразования, сегментация, аппаратное обеспечение
Визуализация информации: проекция и методы параллельных координат

Задачи обучения:
1. Описать основные алгоритмы, лежащие в основе скалярной и векторной визуализации.
2. Описать компромиссные решения в алгоритмах в терминах точности и производительности.
3. Использовать теорию из области обработки сигналов и численного анализа для объяснения эффектов операций визуализации.
4. Описать влияние представления и взаимодействия с пользователем на освоение.

GV10. Виртуальная реальность [факультативный]

Темы:

Стереоскопический дисплей
Имитация обратной связи, осязательные устройства
Определение позиции смотрящего
Обнаружение столкновений
Вычисление видимости
Критичный ко времени рендеринг, несколько уровней детализации (LOD)
Системы виртуальной реальности (VR), основанные на изображениях
Распределенные системы VR, совместная работа с помощью компьютерной сети
Интерактивное моделирование
Проблемы пользовательского интерфейса
Приложения в медицине, моделировании и обучении

Задачи обучения:
1. Описать оптическую модель, реализуемую компьютерной графической системой, для синтеза стереоскопического представления.
2. Описать различные технологии viewer tracking.
3. Объяснить основы алгоритмов эффективного обнаружения коллизий для выпуклых многогранников.

4. Описать отличия между геометрической виртуальной реальностью и виртуальной реальностью, основанной на изображениях.
5. Описать проблемы синхронизации действий пользователя и целостности данных в сетевой среде.
6. Определить основные требования к интерфейсу, аппаратному и программному обеспечению для системы виртуальной реальности, предназначенной для данного приложения.

GV11. Компьютерное зрение [факультативный]

Темы:

Получение изображений
Цифровое изображение и его свойства
Предварительная обработка изображений
Сегментация (установление пороговых значений, сегментация на основе ребер и зон)
Представление формы и распознавание объектов
Анализ движения
Примеры (распознавание объектов, слежение за объектом)

Задачи обучения:

1. Описать процесс формирования изображения.
2. Объяснить преимущества использования двух и более камер, стереоскопическое зрение.
3. Описать различные методы сегментации, их свойства, различия, сильные и слабые стороны.
4. Описать процесс распознавания объекта при помощи представлений, основанных на контурах и зонах.
5. Описать различные методы анализа движения.
6. Описать отличия методов слежения за объектом.

Интеллектуальные системы (IS)

IS1. Основные вопросы, связанные с интеллектуальными системами [обязательный]
IS2. Поиск решений [обязательный]
IS3. Представление знаний и вывод [обязательный]
IS4. Углубленное изучение поиска [факультативный]
IS5. Углубленное изучение представления знаний и вывода [факультативный]
IS6. Агенты [факультативный]
IS7. Обработка естественного языка [факультативный]
IS8. Обучение машины и нейронные сети [факультативный]
IS9. Системы искусственного интеллекта с планируемым поведением [факультативный]
IS10. Робототехника [факультативный]

Область искусственного интеллекта (Artificial Intelligence, AI) связана с проектированием и анализом автономных агентов. Автономные агенты являются программными системами и/или физическими устройствами, оснащенными сенсорами и силовыми приводами, реализованными в виде, например, робота или автономного космического корабля. Интеллектуальная система должна воспринимать изменения в окружающей ее среде, действовать рационально с целью выполнения поставленных перед ней задач, взаимодействовать с другими агентами и с человеком. Эти способности рассматриваются в таких темах, как компьютерное зрение, планирование и поведение, робототехника, мультиагентные системы, распознавание речи и понимание естественного языка. Эти способности основываются на большом наборе общих и специализированных методах представления знаний, механизмах рассуждения и решения задач, алгоритмах поиска и методах машинного обучения.

Кроме того, искусственный интеллект предоставляет средства для решения задач, которые сложно или нерационально решать с помощью других методов. Эти задачи включают в себя эвристический поиск и алгоритмы планирования, формализмы для представления знаний и рассуждения, методы обучения машины, методы очувствления, проблемы распознавания речи и понимания естественного языка, компьютерное зрение, робототехника и другие. Учащийся должен быть в состоянии определить, в каких случаях методы искусственного интеллекта применимы для данной задачи, выбрать походящий метод и реализовать его.

IS1. Основные вопросы, связанные с интеллектуальными системами [обязательный]

Минимальное время, отводимое на раздел: 1 час

Темы:

История искусственного интеллекта
Философские вопросы
– Тест Тьюринга

– Мысленный эксперимент с "китайской комнатой"
– Этические проблемы в искусственном интеллекте
Фундаментальные определения
– Оптимальное рассуждение и человеческое рассуждение
– Оптимальное поведение и человеческое поведение
Моделирование мира
Роль эвристики

Задачи обучения:
1. Описать тест Тьюринга и мысленный эксперимент Сирла с "китайской комнатой".
2. Описать понятия оптимального рассуждения и человеческого рассуждения.
3. Описать понятия оптимального поведения и человеческого поведения.
4. Привести список примеров интеллектуальных систем, которые зависят от моделей мира.
5. Описать роль эвристики и необходимость компромиссного выбора между оптимальностью и эффективностью.

IS2. Поиск решений [обязательный]

Минимальное время, отводимое на раздел: 5 часов

Темы:
Пространство задач
Метод "грубой силы" (поиск в ширину, поиск в глубину, поиск в глубину с итеративным углублением)
Поиск по первому наилучшему совпадению (общий поиск, алгоритм Дейкстры, A*, допустимость A*)
Игры с двумя участниками (минимаксный метод, альфа-бета отсечение)
Поиск допустимого решения (перебор с возвратами и методы локального поиска)

Задачи обучения:
1. Сформулировать эффективное пространство для задачи, поставленной на разговорном языке, выражая это пространство в терминах состояний, операторов, начального состояния и описания целевого состояния.
2. Описать проблему комбинаторного взрыва и ее последствия.
3. Выбрать подходящий алгоритм "грубой силы" для задачи, реализовать его и описать его сложность по времени и по памяти.
4. Выбрать подходящий эвристический алгоритм поиска для задачи и реализовать его, построив эвристическую функцию стоимости.
5. Описать, при каких условиях эвристические алгоритмы дают оптимальное решение.
6. Реализовать минимаксный поиск с альфа-бета отсечением для игры с двумя участниками.
7. Сформулировать проблему, поставленную на разговорном языке, в терминах поиска допустимого решения и реализовать ее решение с помощью хронологического перебора с возвратами.

IS3. Представление знаний и вывод [обязательный]

Минимальное время, отводимое на раздел: 4 часа

Темы:
Обзор логики высказываний и предикатов
Метод резолюций и доказательство теорем
Немонотонный вывод
Вероятностные рассуждения
Теорема Байеса

Задачи обучения:
1. Описать применение метода резолюций в доказательстве теорем.
2. Описать отличия между монотонным и немонотонным выводом.
3. Обсудить преимущества и недостатки вероятностных рассуждений.
4. Научиться применять теорему Байеса для определения условных вероятностей.

IS4. Углубленное изучение поиска [факультативный]

Темы:
Генетический алгоритм
Метод "отжига" (simulated annealing)
Локальный поиск

Задачи обучения:

1. Объяснить понятие генетического алгоритма и сравнить эффективность генетических алгоритмов с классическими методами поиска и решения задач.
2. Показать, как метод "отжига" может использоваться для уменьшения сложности поиска и сравнить этот метод с классическими.
3. Применить локальные методы поиска к классической области.

IS5. Углубленное изучение представления знаний и вывода [факультативный]

Темы:

Структурное представление
– Фреймы и объекты
– Логика описания
– Системы наследования
Немонотонный вывод
– Неклассическая логика
– Рассуждение по умолчанию
– Проверка доверительности
– Логика предпочтений
– Интеграция источников знаний
– Агрегация конфликтующих убеждений
Рассуждение в задачах действия и изменений
– Ситуационное исчисление
– Исчисление событий
– Проблема разветвления
Временное и пространственное мышление
Неопределенности
– Вероятностное рассуждение
– Сети Байеса
– Нечеткие множества и теория вероятностей
– Теория принятия решения
Представление знаний для диагностики и качественное представление

Задачи обучения:

1. Сравнить две распространенные модели структурного представления знаний с указанием их сильных и слабых сторон.
2. Описать составляющие немонотонного рассуждения и их применение в качестве репрезентативных механизмов для доверительных систем
3. Применить ситуационное исчисление и исчисление событий для задач действия и изменений.
4. Описать связи и отличия между временным и пространственным мышлением.
5. Описать основные методы представления неопределенности.
6. Сравнить методы диагностики и качественного представления.

IS6. Агенты [факультативный]

Темы:

Определение агентов
Примеры успешного использования и реальные системы, основанные на агентах
Архитектуры агентов
– Простые реактивные агенты
– Реактивные планировщики
– Многоуровневые архитектуры
– Примеры архитектур и приложений
Теория агентов
– Обязательства
– Намерения
– Агенты, основанные на теории принятия решений
– Марковские процессы принятия решений (MDP)
Программные агенты, персональные помощники, доступ к информации
– Совместно работающие агенты
– Агенты, предназначенные для сбора информации
Правдоподобные агенты (синтетические характеры, моделирование эмоций у агентов)
Самообучающиеся агенты
Мультиагентные системы

– Экономические соображения в пользу мультиагентных систем
– Совместно работающие агенты
– Команды агентов
– Моделирование агентов
– Многоагентное обучение
Введение в теорию роботов-агентов
Мобильные агенты

Задачи обучения:

1. Объяснить, в чем состоят отличия агентов от других категорий интеллектуальных систем.
2. Описать и сравнить стандартные архитектуры агентов.
3. Описать приложения теории агентов к таким областям как программные агенты, персональные помощники и правдоподобные агенты.
4. Описать отличия между обучающимися агентами и агентами, неспособными к обучению.
5. Продемонстрировать на подходящих примерах, как мультиагентные системы поддерживают взаимодействие агентов.
6. Описать и сравнить особенности роботизированных и мобильных агентов.

IS7. Обработка естественного языка [факультативный]

Темы:

Детерминированные и стохастические грамматики
Алгоритмы анализа
Методы, основанные на совокупности текстов
Информационный поиск
Перевод с одного языка на другой
Распознавание речи

Задачи обучения:

1. Дать определение и сравнить детерминированные и стохастические грамматики с примерами, демонстрирующими адекватность грамматик.
2. Описать классические алгоритмы разбора для анализа естественного языка.
3. Объяснить необходимость устоявшейся совокупности текста.
4. Дать примеры каталога и процедур поиска для подхода, основанного на совокупности текста.
5. Описать отличия между методами информационного поиска, перевода с одного языка на другой и распознавания речи.

IS8. Обучение машины и нейронные сети [факультативный]

Темы:

Определение и примеры обучения машины
Обучение под наблюдением
Использование деревьев решений в обучении
Обучаемые нейронные сети
Обучаемые доверительные сети
Алгоритм ближайшего соседа
Теория обучения
Проблема сверхпригодности
Усиленное обучение

Задачи обучения:

1. Объяснить, в чем состоят отличия между тремя основными стилями обучения: обучение под наблюдением, усиленное обучение и безнадзорное обучение.
2. Реализовать простые алгоритмы обучения под наблюдением, усиленного обучения и безнадзорного обучения.
3. Определить, какой из трех стилей обучения применим для решения данной задачи.
4. Сравнить каждый из следующих методов, с приведением примеров, в которых метод дает наилучшие результаты: деревья решений, нейронные сети и доверительные сети.
5. Реализовать простую обучающуюся систему с использованием деревьев решений, нейронных сетей и/или доверительные сети, выбирая наиболее подходящий из этих методов.
6. Охарактеризовать современное положение дел в теории обучения, включая ее достижения и недостатки.
7. Объяснить алгоритм ближайшего соседа и его место в теории обучения.
8. Объяснить проблему сверхпригодности, методы обнаружения этой проблемы и управления ей.

IS9. Системы искусственного интеллекта с планируемым поведением [факультативный]

Темы:

Определения и примеры систем с планируемым поведением
Планирование как поиск
Планирование с участием оператора
Пропозициональное планирование
Расширенные системы планирования (системы планирования, основанные на случаях использования, обучающиеся и вероятностные системы)
Статические системы планирования
Планирование и исполнение
Планирование и робототехника

Задачи обучения:

1. Определить понятие системы с планируемым поведением.
2. Объяснить, в чем состоят отличия системы искусственного интеллекта с планируемым поведением от классических методов поиска.
3. Описать отличия между планированием как поиском решения, планированием с участием оператора и пропозициональным планированием, с указанием предметных областей, в которых применим каждый из этих методов.
4. Описать каждый из следующих методов: планирование, основанное на случаях использования, планирование с обучением и вероятностное планирование.
5. Сравнить статические системы планирования с системами, в которых требуется динамическое выполнение.
6. Описать влияние динамического планирования на робототехнику.

IS10. Робототехника [факультативный]

Темы:

Обзор
– Реально использующиеся робототехнические системы
– Планирование в сравнении с реактивным управлением
– Неопределенность в управлении
– Очувствление (sensing)
– Модели мира
Пространство конфигураций
Планирование
Очувствление
Подготовка программ для роботов
Навигация и управление

Задачи обучения:

1. Обрисовать потенциал и ограничения современных робототехнических систем, использующихся на практике.
2. Реализовать алгоритмы пространства конфигураций для двумерного робота и сложных многоугольников.
3. Реализовать простые алгоритмы планирования движения.
4. Описать неопределенности, связанные с датчиками, и методы борьбы с этими неопределенностями.
5. Спроектировать простую архитектуру управления.
6. Описать различные стратегии для навигации в неизвестных средах, достоинства и недостатки каждой из них.
7. Описать различные стратегии для навигации с помощью ориентиров, достоинства и недостатки каждой из них.

Управление информацией (IM)

IM1. Информационные модели и системы [обязательный]
IM2. Системы баз данных [обязательный]
IM3. Моделирование данных [обязательный]
IM4. Реляционные базы данных [факультативный]
IM5. Языки запросов к базам данных [факультативный]
IM6. Проектирование реляционных баз данных [факультативный]
IM7. Обработка транзакций [факультативный]
IM8. Распределенные базы данных [факультативный]
IM9. Проектирование физической структуры базы данных [факультативный]

IM10. Информационная проходка [факультативный]
IM11. Хранение и поиск информации [факультативный]
IM12. Гипертекст и гипермедиа [факультативный]
IM13. Мультимедийная информация и системы мультимедиа [факультативный]
IM14. Цифровые библиотеки [факультативный]

Управление информацией (IM) играет важнейшую роль практически во всех областях, где используются компьютеры. Этот раздел включает в себя такие вопросы, как сбор информации, перевод ее в цифровую форму, представление, организация, преобразование и выдача информации, алгоритмы для эффективного доступа и обновления хранимой информации, моделирования данных и абстракции, а также методы физического хранения информации. Помимо этого, в данном разделе изучаются проблемы безопасности, конфиденциальности, целостности и защиты разделяемых данных. Учащийся должен уметь разрабатывать концептуальную и физическую модели данных, определять, какие методы управления информацией подходят для задачи, уметь выбрать и реализовать подходящее решение, удовлетворяющее всем требованием, включая масштабируемость и удобство использования.

IM1. Информационные модели и системы [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

История и причины появления информационных систем
Хранение и поиск информации (Information Storage & Retrieval)
Приложения теории управления информацией
Получение и представление информации
Анализ и индексирование
Поиск, извлечение данных, связывание, навигация
Конфиденциальность, целостность, безопасность и сохранность информации
Масштабируемость, производительность и эффективность

Задачи обучения:

1. Сравнить информацию с данными и знаниями.
2. Описать эволюцию информационных систем, начиная с ранних взглядов в этой области и заканчивая современными системами, с указанием их возможностей и потенциала для развития.
3. Сравнить информационные системы малого и среднего размера с точки зрения удовлетворения потребностей реального пользователя.
4. Описать некоторые технические решения проблем, связанных с конфиденциальностью, целостностью, безопасностью и сохранностью информации.
5. Объяснить вопросы производительности (пропускная способность, время отклика) и эффективности (восстановление, точность).
6. Описать методы, позволяющие проверить масштабируемость системы.

IM2. Системы баз данных [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

История и причины появления систем баз данных
Компоненты системы баз данных
Функции систем управления базами данных (СУБД)
Архитектура базы данных и независимость данных
Использование языка запросов к системе баз данных

Задачи обучения:

1. Описать черты, отличающие использование баз данных от традиционного подхода к программированию с файлами данных.
2. Выделить основные цели, функции, модели, компоненты, приложения и социальные аспекты систем баз данных.
3. Описать компоненты системы баз данных и дать примеры их использования.
4. Описать основные функции СУБД и их роль в системах баз данных.
5. Объяснить понятие независимости данных и его важность в системах баз данных.
6. Научиться использовать язык запросов для извлечения информации из базы данных.

IM3. Моделирование данных [обязательный]

Минимальное время, отводимое на раздел: 4 часа

Темы:

Моделирование данных
Концептуальные модели (включая модель "объект-отношение" и язык UML)
Объектно-ориентированная модель
Реляционная модель данных

Задачи обучения:

1. Классифицировать модели данных на основании понятий, с помощью которых они описывают структуру базы данных – концептуальная модель данных, физическая модель данных и репрезентативная модель данных.
2. Описать понятие моделирования, нотацию модели "объект-отношение" и языка UML, включая использование этих моделей в моделировании данных.
3. Описать основные понятия объектно-ориентированной модели: идентификатор объекта, конструкторы типов, инкапсуляция, наследование, полиморфизм и версионирование.
4. Определить основные термины реляционной модели данных.
5. Проиллюстрировать понятия моделирования и нотацию реляционной модели данных.

IM4. Реляционные модели данных [факультативный]

Темы:

Отображение концептуальной схемы в реляционную схему
Целостность сущностей-объектов и ссылочная целостность
Реляционная алгебра и реляционное исчисление

Задачи обучения:

1. Создать реляционную схему из концептуальной схемы, разработанной с помощью модели "объект-отношение".
2. Объяснить и показать на примере понятие ограничений целостности объектов и ссылочной целостности (включая определение понятия внешнего ключа).
3. Показать на примерах использование операций реляционной алгебры из математической теории множеств (*объединение, пересечение, разность,* и *прямое произведение*) и операций реляционной алгебры, разработанных специально для реляционных баз данных (*выбока, произведение, соединение* и *деление*).
4. Продемонстрировать использование запросов в реляционной алгебре.
5. Продемонстрировать использование запросов в реляционном исчислении кортежей.

IM5. Языки запросов к базам данных [факультативный]

Темы:

Обзор языков, применяемых в базах данных
Язык SQL (определение данных, написание запросов, команда обновления, ограничения, целостность)
Оптимизация запросов
Запросы по образцу и среды четвертого поколения
Встраивание непроцедурных запросов в процедурный язык
Введение в объектно-ориентированный язык запросов (Object Query Language)

Задачи обучения:

1. Создать на SQL схему реляционной базы данных, включающую ключи, ограничения целостности сущностей-объектов и ограничения ссылочной целостности.
2. Продемонстрировать определение данных на SQL и извлечение данных из базы с помощью оператора SQL SELECT.
3. Оценить несколько стратегий обработки запроса и выбрать из них оптимальную.
4. Создать непроцедурный запрос путем заполнения шаблонов отношений для построения примера требуемого результата запроса.
5. Встроить объектно-ориентированные запросы в язык программирования, такой как C++ или Java (например, SELECT Col.Method() FROM Object).

IM6. Проектирование реляционных баз данных [факультативный]

Темы:

Проектирование базы данных
Функциональная зависимость
Нормальные формы (1НФ, 2НФ, 3НФ, НФБК)
Многозначная зависимость (4НФ)

Зависимости соединения и пятая нормальная форма (5НФ)
Теория представлений

Задачи обучения:

1. Определить, есть ли функциональная зависимость между двумя или более атрибутами, являющимися подмножеством отношения.
2. Описать понятия 1НФ, 2НФ, 3НФ и НФБК (нормальная форма Бойса-Кодда).
3. Определить, находится ли отношение в одной из нормальных форм: 1НФ, 2НФ, 3НФ или НФБК.
4. Разделить отношение, находящееся в 1НФ, на несколько отношений, находящихся в 3НФ (или НФБК), и денормализовать реляционную схему.
5. Объяснить влияние нормализации на эффективность операций с базой данных, особенно на оптимизацию запросов.
6. Объяснить понятие многозначной зависимости и описать ограничения, налагаемые такой зависимостью.
7. Объяснить, почему 4НФ полезна при проектировании схемы.

IM7. Обработка транзакций [факультативный]

Темы:

Транзакции
Сбой и восстановление
Управление параллелизмом

Задачи обучения:

1. Инициировать транзакцию с помощью запроса на SQL, встроенного в приложение.
2. Объяснить понятие неявной фиксации (транзакции).
3. Описать проблемы эффективного выполнения транзакций.
4. Описать, когда и почему необходим откат транзакции и как регистрация гарантирует правильный откат.
5. Объяснить воздействие уровней изоляции на механизмы параллелизма.
6. Выбрать правильный уровень изоляции для реализации заданного протокола транзакции.

IM8. Распределенные базы данных [факультативный]

Темы:

Распределенные хранилища данных
Обработка распределенных запросов
Модель распределенных транзакций
Управление параллелизмом
Гомогенные и гетерогенные решения
Клиент-сервер

Задачи обучения:

1. Описать методы, используемые для фрагментации данных, репликации и размещения в процессе проектирования распределенной базы данных.
2. Оценить простые стратегии выполнения распределенного запроса с целью выбора стратегии, минимизирующей количество пересылаемых данных.
3. Объяснить, как протокол двухфазного завершения используется для фиксации транзакции, затрагивающей данные, находящиеся на нескольких узлах сети.
4. Описать управление распределенным параллелизмом на основании методов выделенной копии и метода голосования.
5. Описать три уровня программного обеспечения в модели "клиент-сервер".

IM9. Проектирование физической структуры базы данных [факультативный]

Темы:

Хранилище и файловая структура
Индексированные файлы
Хешированные файлы
Файлы сигнатур
B-деревья
Файлы с плотными индексами
Файлы с записями переменной длины
Эффективность и настройка базы данных

Задачи обучения:

1. Объяснить понятия записи, типа записи, файла; описать различные способы размещения файловых записей на диске.
2. Привести примеры использования первичных, вторичных и кластерных индексов.
3. Описать отличия между неплотными и плотными индексами.
4. Реализовать динамические многоуровневые индексы с использованием B-деревьев.
5. Описать теорию и приложения методов внутреннего и внешнего хеширования.
6. Применить хеширование для управления динамическим расширением файла.
7. Описать связь между хешированием, сжатием и эффективным поиском в базе данных.
8. Оценить накладные расходы и преимущества различных схем хеширования.
9. Объяснить влияние физической структуры базы данных на эффективность транзакций.

IM10. Извлечение информации [факультативный]

Темы:

Полезность извлечения информации
Ассоциативные и последовательные шаблоны
Кластеризация данных
Анализ потребительской корзины
Очистка данных
Визуализация данных

Задачи обучения:

1. Сравнить различные концепции извлечения информации, используемые в исследованиях и приложениях.
2. Объяснить роль поиска ассоциаций в коммерческих данных потребительской корзины.
3. Охарактеризовать виды шаблонов, обнаруживаемые поиском ассоциативных правил.
4. Описать, каким образом можно расширить реляционную систему для поиска шаблонов с использованием правил ассоциации.
5. Оценить методологические вопросы, лежащие в основе эффективного использования извлечения информации.
6. Указать и охарактеризовать источники "шума", избыточности и посторонних значений в данных.
7. Указать механизмы замыкания контура при извлечении информации (агрегирование "на лету", интерактивная визуализация).
8. Объяснить, почему различные процессы замыкания контура улучшают эффективность извлечения информации.

IM11. Хранение и поиск информации [факультативный]

Темы:

Символы, строки, кодирование, текст
Документы, электронная публикация, разметка и языки разметки
Деревья, предметные указатели, PAT-деревья, файлы сигнатур, индексирование
Морфологический анализ, анализ разветвлений, фразы, "черные списки"
Плотности распределения термов, неопределенность, нечеткость, взвешивание
Векторное пространство, вероятностная, логическая и расширенная модели
Информационные потребности, релевантность, оценка, эффективность
Тезаурус, онтология, классификация и категоризация, метаданные
Библиографическая информация, библиометрика, цитаты
Маршрутизация и фильтрация
Поиск и стратегия поиска, поведение алгоритма поиска информации, моделирование пользователя, обратная связь
Резюмирование и визуализация информации
Интеграция цитирования, ключевых слов, схемы классификации и другие термины
Протоколы и системы (включая Z39.50, системы OPAC, поисковые WWW-машины, исследовательские системы)

Задачи обучения:

1. Объяснить базовые понятия хранения и поиска информации.
2. Описать проблемы, специфические для эффективного поиска информации.
3. Привести примеры альтернативных стратегий поиска и объяснить, почему данная стратегия применима в данном случае.
4. Выполнить исследование на основе материалов из Internet.
5. Спроектировать и реализовать систему хранения и поиска информации малого или среднего размера.

IM12. Гипертекст и гипермедиа [факультативный]

Темы:

Модели гипертекста (исторические первые модели, web, Dexter, Amsterdam, HyTime)
Сервисы связи, машины, и (распределенные) архитектуры гипертекста
Узлы, композиты и якоря (anchors)
Измерения, единицы измерения, размещения, интервалы
Просмотр, навигация, виды, масштабирование
Автоматическая генерация ссылок
Представление, преобразование, синхронизация
Разработка, чтение и аннотирование
Протоколы и системы (включая web, HTTP)

Задачи обучения:

1. Резюмировать эволюцию моделей гипертекста и гипермедиа от первых вариантов до современного состояния, указывая их возможности и ограничения.
2. Объяснить базовые понятия гипертекста и гипермедиа.
3. Продемонстрировать понимание основ представления, преобразования и синхронизации информации.
4. Сравнить доставку гипермедиа информации на основе различных протоколов и систем.
5. Спроектировать и реализовать с помощью инструментальных средств разработки приложение для поиска информации, пригодное для использования в среде Web.

IM13. Мультимедийная информация и системы мультимедиа [факультативный]

Темы:

Устройства, драйверы устройств, сигналы и протоколы управления, системы DSP
Приложения, редакторы мультимедиа-информации, авторские системы и авторская разработка
Потоки/структуры, захват/представление/преобразование, пространства/области, сжатие/кодирование
Анализ на основе содержимого, индексирование и поиск аудио-, видео-информации и изображений
Представление, рендеринг, синхронизация, мультимодальная интеграция/интерфейсы
Доставка данных в режиме реального времени, качество обслуживания, аудио/видео конференции, видео по запросу

Задачи обучения:

1. Описать устройства мультимедиа и вспомогательные устройства, связанные с мультимедиа-информацией и системами.
2. Объяснить базовые концепции представления информации с использованием средств мультимедиа.
3. Продемонстрировать использование анализа информации на основе содержимого в информационных мультимедиа-системах.
4. Критически оценить мультимедиа-презентации в терминах правильного использования аудио, видео, графики, цвета и других средств представления информации.
5. Реализовать мультимедиа-приложение с использованием коммерческой среды разработки.

IM14. Цифровые библиотеки [факультативный]

Темы:

Оцифровка, хранение и обмен
Цифровые объекты, композиты и пакеты
Метаданные, каталогизация, подача документов автором
Именование, репозитории, архивы
Пространства (концептуальные, географические, 2/3D, VR)
Архитектуры (агенты, шины, обертки/посредники), взаимодействие
Сервисы (поиск, связывание, просмотр и так далее)
Управление правами на интеллектуальную собственность, конфиденциальность, защита ("водяные знаки")
Архивация и сохранение, целостность

Задачи обучения:

1. Описать технические концепции, лежащие в основе создания цифровых библиотек.
2. Описать основные требования к поиску, связыванию и просмотру.
3. Обсудить сценарии, включающие примеры правильного и неправильного использования цифровой библиотеки, социальные, правовые и экономические последствия каждого сценария.
4. Описать некоторые технические решения проблем, относящихся к архивации и сохранности информации в цифровой библиотеке.
5. Спроектировать и реализовать небольшую цифровую библиотеку.

Социальные и профессиональные вопросы (SP)

SP1. История информатики [обязательный]
SP2. Социальный контекст информатики [обязательный]
SP3. Методы и средства анализа [обязательный]
SP4. Профессиональная и этическая ответственность [обязательный]
SP5. Недостатки компьютерных систем и риски, связанные с их применением [обязательный]
SP6. Интеллектуальная собственность [обязательный]
SP7. Конфиденциальность и гражданские свободы [обязательный]
SP8. Компьютерные преступления [факультативный]
SP9. Экономические вопросы, связанные с применением компьютеров [факультативный]
SP10. Философские вопросы [факультативный]

Технические вопросы являются основой любой учебной программы в области информатики, однако нельзя ими ограничиваться. Учащиеся должны также понимать социальный и профессиональный контекст информатики (social and professional issues).

Необходимость включения социальных вопросов в документ со всей очевидностью выражена в следующей выдержке из CC1991 [Tucker91]:

> Студенты должны понимать основные культурные, социальные, правовые и этические аспекты информатики. Они должны понимать прошлое информатики, ее текущее состояние и направления развития. Они должны сознавать свою личную роль в этом процессе, понимать философские вопросы, технические проблемы и эстетические ценности, которые играют важную роль в развитии дисциплины.
>
> Студенты должны также развивать в себе способность задавать серьезные вопросы о социальном влиянии информатики и оценивать предлагаемые ответы на них. Будущие практики должны уметь предвидеть последствия внедрения данного продукта в данную среду. Будет ли этот продукт улучшать качество жизни, или наоборот, ухудшит его? Каким будет его воздействие на отдельных людей, группы и организации?
>
> Наконец, студенты должны знать основные правовые нормы, касающиеся производителей и пользователей программного и аппаратного обеспечения, и сознавать этические ценности, лежащие в основе этих норм. Будущие практики должны понимать ответственность, которая ляжет на них, и возможные последствия неудачи. Они должны понимать, что их собственные возможности небезграничны, как и возможности используемых ими инструментальных средств. Все практики должны взять на себя обязательство идти в ногу как с развитием областей, в которых они специализируются, так и с развитием всей информатики в целом.

Материал данной дисциплины лучше всего преподавать в виде обязательного курса и небольших разделов в других курсах. С одной стороны, некоторые разделы, помеченные как обязательные – в частности, SP2, SP3, SP4 и SP6 – не слишком подходят для изучения в других традиционных курсах, так как без специального курса эти разделы сложно изучить в необходимом объеме. С другой стороны, представление этических вопросов лишь в отдельном курсе, без контекста, может создать ложное представление о том, что технические процессы не несут с собой этических проблем. Поэтому важно, чтобы некоторые традиционные курсы включали в себя разделы, в которых этические вопросы анализируются в контексте технической темы курса. Курсы в таких областях, как разработка программного обеспечения, базы данных, компьютерные сети и введение в информатику, обеспечивают подходящий контекст для рассмотрения этических проблем. Однако, материалы, касающиеся этических вопросов, можно включить практически в любой курс в учебной программе. Ограничиться только одним отдельным курсом, посвященным этим вопросам, было бы очевидным противоречием духу рекомендаций документа. Перечисление всех возможных проблем в данной области призвано дать практикам импульс активно бороться с этими проблемами, как морально, так и технически.

Этические вопросы, обсуждаемые на любом занятии, должны естественным образом возникать из темы занятия и иметь отношение к этой теме. Среди примеров можно назвать обсуждение агрегирования данных или извлечения информации в курсе, посвященном базам данных, или обсуждение возможных конфликтов между гарантиями по отношению к заказчику и гарантиями по отношению к пользователям и другим людям, на которых отразится работа пользователей, в курсе разработки программного обеспечения. Задания по программированию, построенные вокруг таких приложений, как управление перемещением лазера во время глазной операции, могут помочь увидеть профессиональные, этические и социальные аспекты информатики.

Имеются разногласия педагогического характера в отношении периода обучения, на котором следует читать обязательный курс в данной области: на первом-втором годах обучения или на предпоследнем-выпускном. Чтение курса на первых годах обучения:

1. Позволяет изучить методы и средства анализа (SP3) до перехода к анализу этических проблем в контексте различных технических областей.
2. Гарантирует, что студенты, которые рано бросят учебу и начнут работать, будут иметь базовые представления о профессиональных и этических вопросах.

С другой стороны, чтение курса на ранних этапах обучения может привести к одной из следующих проблем:

1. Студенты первых лет обучения могут не иметь технических знаний и интеллектуальной зрелости для глубокого анализа этических проблем. Без понимания технических альтернатив, трудно рассматривать их этические последствия.
2. Студенты должны быть достаточно зрелыми и опытными, чтобы оценить предпосылки возникновения проблем. Для этого студенты должны закончить, по крайней мере, вводный курс в области дискретной математики и второй курс в области информатики. Кроме того, если в учебной программе присутствует курс по написанию технической документации, то этот курс должен быть обязательным предварительным условием для курса в области SP.
3. Авторы некоторых учебных программ могут захотеть использовать этот курс как заключительный для студентов последних лет обучения.

Хотя разделы SP2 и SP3 приведены вместе с количеством часов, отводимых для них, эти разделы являются фундаментальными по отношению ко всем остальным. Поэтому, при изучении других разделов, преподаватели должны все время иметь в виду вопросы социального контекста и развитие навыков этического анализа. На практике это означает, что темы из разделов SP2 и SP3 должны постоянно пополняться материалом из других областей по мере их изучения.

SP1. История информатики [обязательный]

Минимальное время, отводимое на раздел: 1 час

Темы:

Предыстория – мир до 1946 года
История аппаратного и программного обеспечения компьютеров, компьютерных сетей
Пионеры информатики

Задачи обучения:

1. Описать вклад пионеров информатики в данную область.
2. Сравнить повседневную жизнь до и после появления персональных компьютеров и Internet.
3. Упомянуть важные тенденции в истории развития информатики.

SP2. Социальный контекст информатики [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Введение в социальные последствия информатики
Социальные последствия компьютерных сетей
Рост, управление и доступ к Internet
Вопросы пола
Международные вопросы

Задачи обучения:

1. Проинтерпретировать социальный контекст конкретной реализации какого-либо проекта.
2. Указать предположения и ценности, заключенные в данном дизайне.
3. Оценить данную реализацию путем использованием эмпирических данных.
4. Описать позитивные и негативные стороны расширения взаимодействия между людьми с помощью компьютеров.
5. Объяснить, почему в некоторых странах ограничен доступ к компьютерам и компьютерным сетям.

SP3. Методы и средства анализа [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Выдвижение и оценка этических аргументов
Выявление и оценка альтернатив с этической точки зрения
Понимание социального контекста проектирования
Выявление предположений и ценностей

Задачи обучения:
1. Научиться анализировать аргументы с целью выявления исходных посылок и вывода.
2. Проиллюстрировать использование примера, аналогии и контр-аналогии в качестве этического аргумента.
3. Научиться обнаруживать применение логических обманов в качестве аргумента.
4. Выявить заинтересованные в проблеме стороны и наши ответственности перед ними.
5. Связать этические компромиссы с техническим решением.

SP4. Профессиональные и этические ответственности [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Общественные ценности и законы, по которым мы живем
Природа профессионализма
Различные формы профессиональной идентификации, их достоинства и недостатки
Роль профессионализма в общественной политике
Необходимость осознания последствий
Расхождение этических взглядов и сигнализация разногласий
Кодексы этики, поведения и практики (IEEE, ACM, SE, AITP и др.)
Борьба с притеснением и дискриминацией
Политика "допустимого использования" компьютеров на рабочем месте

Задачи обучения:
1. Описать последовательные этапы развития инцидента с возникновением этических конфликтов.
2. Указать сильные и слабые стороны различных профессиональных кодексов как выражений профессионализма и руководств к принятию решений.
3. Описать этические проблемы, возникающие при разработке программного обеспечения и пути их решения, как с этической точки зрения, так и в техническом плане.
4. Разработать политику использования компьютера, включая меры обеспечения этой политики.
5. Проанализировать глобальную компьютерную проблему, рассмотреть роли профессионалов и правительственных чиновников в этом вопросе.
6. Оценить профессиональные кодексы этики ACM, IEEE Computer Society и других организаций.

SP5. Недостатки компьютерных систем и риски, связанные с их применением [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Исторические примеры рисков, связанных с программным обеспечением (например, случай с Therac-25)
Следствия сложности программного обеспечения
Оценка рисков и управление ими

Задачи обучения:
1. Описать ограничения тестирования как средства проверки корректности.
2. Описать различия между корректностью, надежностью и безопасностью.
3. Обсудить возможные неочевидные проблемы, связанные с повторным использованием существующих компонент.
4. Обсудить современные подходы к управлению рисками, охарактеризовать сильные стороны и недостатки каждого из них.

SP6. Интеллектуальная собственность [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Основы интеллектуальной собственности
Авторские права, патенты и коммерческие тайны
Нарушение авторских прав на программное обеспечение
Патенты на программное обеспечение
Международные вопросы, касающиеся интеллектуальной собственности

Задачи обучения:

1. Описать отличия между патентом, авторским правом и коммерческой тайной.
2. Обсудить правовые основы авторских прав в национальном и международном законах.
3. Объяснить возможные отличия между законами разных стран с точки зрения защиты патентов и авторских прав.
4. Обрисовать историческое развитие патентов на программное обеспечение.
5. Обсудить последствия нарушения авторских прав на программное обеспечение для разработчиков и роль соответствующих правоохранительных организаций.

SP7. Конфиденциальность и гражданские свободы [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Этические и правовые основы обеспечения конфиденциальности (privacy protection)
Последствия использования больших баз данных с точки зрения конфиденциальности
Технологические стратегии обеспечения конфиденциальности
Свобода выражения в киберпространстве
Межнациональные и межкультурные последствия

Задачи обучения:

1. Описать правовые основы обеспечения конфиденциальности и свободы самовыражения в своей стране и отличия этих понятий в различных странах.
2. Описать угрозы конфиденциальности, возникающие в связи с применением компьютеров.
3. Объяснить, как Internet может нарушить исторический баланс в защите свободы самовыражения.
4. Описать преимущества и недостатки свободы самовыражения в киберпространстве.
5. Описать тенденции в обеспечении конфиденциальности на технологических примерах.

SP8. Компьютерные преступления [факультативный]

Темы:

История и примеры компьютерных преступлений
"Взлом" ("хакерство") и его эффекты
Вирусы, черви и троянские кони
Стратегии предотвращения преступлений

Задачи обучения:

1. Обрисовать технические основы вирусов и атак, направленных на вызов отказа в обслуживании.
2. Перечислить методы борьбы с атаками, направленными на "взлом" (cracking).
3. Обсудить различные подходы и мотивации к проблеме "взлома".
4. Указать роль профессионалов в обеспечении безопасности и компромиссы в решениях проблем безопасности.

SP9. Экономические вопросы, связанные с применением компьютеров [факультативный]

Темы:

Монополии и их экономические последствия
Нехватка квалифицированной рабочей силы и качество компьютерных продуктов
Ценовые стратегии в области компьютеров
Различия в доступе к вычислительным ресурсам и возможные последствия этих различий

Задачи обучения:

1. Объяснить основные причины борьбы с монополиями.
2. Описать, как нехватка квалифицированной рабочей силы влияет на индустрию информационных технологий.
3. Предложить и обсудить пути преодоления ограничений доступности вычислительных ресурсов.
4. Обрисовать эволюцию ценовых стратегий для компьютерных товаров и услуг.

SP10. Философские вопросы [факультативный]

Темы:

Философские течения, в частности, утилитаризм и деонтология (теория проблем долга и моральных требований)

Проблемы этического релятивизма
Научная этика в исторической перспективе
Различия в научных и философских подходах

Задачи обучения:
1. Описать основные концепции релятивизма, утилитаризма и деонтологии.
2. Выделить отличия между этической теорией и профессиональной этикой.
3. Указать слабые стороны следующих этических концепций: "сотрудник по найму", строгий легализма, наивный эгоизм, наивный релятивизм.

Программная инженерия (SE)

SE1. Проектирование ПО [обязательный]
SE2. Использование программных интерфейсов приложений [обязательный]
SE3. Программные средства и окружения [обязательный]
SE4. Процессы разработки ПО [обязательный]
SE5. Спецификации и требования к ПО [обязательный]
SE6. Проверка соответствия ПО [обязательный]
SE7. Эволюция ПО [обязательный]
SE8. Управление программными проектами [обязательный]
SE9. Компонентно-ориентированная разработка [факультативный]
SE10. Формальные методы [факультативный]
SE11. Надежность ПО [факультативный]
SE12. Разработка специализированных систем [факультативный]

Программная инженерия (software engineering) – это дисциплина, рассматривающая приложение теории, знаний и практики с целью эффективного построения программных систем, удовлетворяющих требованиям пользователей и клиентов. Эта дисциплина находит приложения в разработке систем любого масштаба – от небольших до самых крупных. Данная дисциплина изучает все фазы жизненного цикла программной системы: анализ требований, разработку спецификаций, проектирование, построение, тестирование, эксплуатацию и сопровождение.

Разработка программного обеспечения использует инженерные методы, процессы, техники и измерения. Разработка ПО извлекает пользу из применения средств для управления разработкой, анализа и моделирования программных продуктов; оценки и контроля за качеством; дисциплинированного и управляемого подхода к эволюции программного обеспечения и повторного использования. Разработка программного обеспечения как силами одного разработчика, так и коллектива, требует применения инструментальных средств, методов и подходов, которые лучше всего подходят для данной среды разработки.

Элементы данной дисциплины применимы в любых предметных областях, связанных с использованием компьютеров, где профессионализм, качество, планирование и стоимость важны при производстве программных систем.

SE1. Проектирование ПО [обязательный]

Минимальное время, отводимое на раздел: 8 часов

Темы:
Фундаментальные концепции и принципы проектирования
Шаблоны проектирования (design patterns)
Архитектура ПО
Структурное проектирование
Объектно-ориентированный анализ и проектирование
Проектирование на уровне компонентов
Проектирование с учетом повторного использования

Задачи обучения:
1. Обсудить свойства проектирования хорошего ПО.
2. Сравнить объектно-ориентированный анализ и проектирование со структурным анализом и проектированием.
3. Оценить качество нескольких программных проектов на основе ключевых принципов и концепций проектирования.
4. Выбрать и применить шаблон проектирования, подходящий для разработки данного приложения.

5. Спроектировать программный продукт среднего размера с использованием спецификации требований к программному обеспечению, одной из методологий проектирования (структурной или объектно-ориентированной) и соответствующей нотации.
6. Провести обзор архитектуры программного проекта, используя соответствующие рекомендации.
7. Оценить программный проект на уровне компонент.
8. Оценить программный проект с точки зрения повторного использования.

SE2. Использование программных интерфейсов приложений [обязательный]

Минимальное время, отводимое на раздел: 5 часов

Темы:

Программирование с использованием API
Браузеры классов и подобные средства
Программирование с помощью примеров
Отладка при использовании API
Введение в программирование на основе компонентов

Задачи обучения:

1. Объяснить значение программных интерфейсов приложений (API) в разработке ПО.
2. Использовать браузеры классов и подобные средства в процессе разработки приложений с использованием API.
3. Спроектировать, реализовать, протестировать и отладить программы, использующие большие пакеты API.

SE3. Программные средства и окружения [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Среды программирования
Анализ требований и средства моделирования проекта
Инструментальные средства тестирования
Инструментальные средства управления конфигурацией
Механизмы интеграции инструментальных средств

Задачи обучения:

1. Выбрать и обосновать набор инструментальных средств для поддержки разработки ряда программных продуктов.
2. Проанализировать и оценить набор инструментальных средств в заданной области разработки ПО (например, управление проектом, моделирование или тестирование).
3. Продемонстрировать навыки в использовании инструментальных средств для поддержки разработки программного продукта среднего размера.

SE4. Процессы разработки ПО [обязательный]

Минимальное время, отводимое на раздел: 2 часа

Темы:

Жизненный цикл ПО и модели процесса разработки
Модели оценки процесса разработки
Метрики процесса разработки ПО

Задачи обучения:

1. Описать жизненный цикл ПО, его этапы и результаты каждого этапа.
2. Выбрать (с обоснованием выбора) модели разработки ПО, наиболее подходящие для разработки и сопровождения нескольких несхожих программных продуктов.
3. Объяснить роль моделей зрелости процесса разработки.
4. Сравнить традиционную водопадную модель с инкрементальной моделью, объектно-ориентированной моделью и др.
5. Для каждого из нескольких различающихся сценариев программного проекта описать место проекта в жизненном цикле продукта, выявить задачи, которые должны быть выполнены на следующем этапе, и указать метрику, подходящую для таких задач.

SE5. Спецификации и требования к ПО [обязательный]

Минимальное время, отводимое на раздел: 4 часа

Темы:

Выявление требований
Методы моделирования и анализа требований
Функциональные и нефункциональные требования
Прототипирование
Основные концепции методов формальной спецификации

Задачи обучения:

1. Применить ключевые элементы и типичные методы выявления и анализа требований для построения набора требований к программной системе среднего размера.
2. Обсудить задачу поддержки устаревшей программной системы.
3. Использовать распространенные неформальные методы для моделирования и спецификации в виде технического задания требований для программной системы среднего размера.
4. Провести просмотр технического задания для определения качества документа.
5. Перевести на естественный язык техническое задание, написанное на одном из известных языков формальных спецификаций.

SE6. Проверка соответствия ПО [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Планирование проверки соответствия (validation planning)
Основы тестирования, включая создание тестового плана и генерацию тестов
Тестирование методом "черного ящика" и "белого ящика"
Тестирование элементов, интеграционное, системное тестирование и проверка соответствия
Объектно-ориентированное тестирование
Проверки равных по рангу и инспекции (peer reviews, inspections)

Задачи обучения:

1. Описать отличия между проверкой правильности (verification) и проверкой соответствия (validation).
2. Описать роль инструментальных средств в проверке соответствия программного обеспечения.
3. Описать отличия между различными типами и уровнями тестирования (тестирование модулей, интеграционное тестирование, системное тестирование и приемо-сдаточное тестирование) для программных продуктов среднего размера.
4. Создать, оценить и реализовать тестовый план для модуля кода среднего размера.
5. Провести, в качестве одной из составляющих работы команды, совместный просмотр модуля кода среднего размера.
6. Обсудить вопросы, связанные с тестированием объектно-ориентированного ПО.

SE7. Эволюция ПО [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Сопровождение ПО
Свойства ПО, пригодного для сопровождения
Реинжиниринг
Унаследованные или устаревшие системы (legacy systems)
Повторное использование ПО

Задачи обучения:

1. Описать основные проблемы, связанные с эволюцией ПО и их влияние на жизненный цикл ПО.
2. Обсудить проблемы сопровождения унаследованных систем и необходимость возвратного проектирования (reverse engineering).
3. Обрисовать процесс регрессионного тестирования и его роль в управлении выпуском версий продукта.
4. Оценить последствия запроса на изменение для существующего продукта среднего размера.
5. Разработать план реинжиниринга продукта среднего размера в ответ на запрос на изменение.
6. Обсудить преимущества и недостатки повторного использования программного обеспечения.
7. Рассмотреть возможности повторного использования ПО в заданно контексте.

SE8. Управление программными проектами [обязательный]

Минимальное время, отводимое на раздел: 3 часа

Темы:

Управление группой
– Процессы, протекающие в коллективе
– Организация группы и принятие решений
– Роли и ответственности в группе разработчиков
– Выявление и назначение ролей
– Слежение за состоянием проекта
– Решение проблем коллектива
Планирование проекта
Методы оценки и измерения ПО
Анализ рисков
Обеспечение качества ПО
Управление конфигурацией программного продукта
Инструментальные средства управления проектом

Задачи обучения:

1. Продемонстрировать основные принципы создания команды и управления коллективом на примере группового проекта.
2. Создать проектный план для программного проекта, включающий оценку размера и трудозатрат, график проекта, распределение ресурсов, управление конфигурацией, управление изменениями, а также выявление рисков, связанных с проектом и управление ими.
3. Сравнить различные методы обеспечения качества программного продукта.

SE9. Компонентно-ориентированная разработка [факультативный]

Темы:

Основы
– Определение и свойства компонентов
– Компоненты и интерфейсы
– Интерфейс как контракт
– Выгоды, которые дает применение компонентов
Основные методы
– Разработка и сборка компонентов
– Соотношение с моделью "клиент-сервер" и шаблонами
– Использование объектов и сервисы, предоставляемые объектами за время своего жизненного цикла
– Использование объектных брокеров
– Маршаллинг
Приложения (включая использование мобильных компонент)
Архитектура систем, основанных на компонентах
Компонентно-ориентированное проектирование
Обработка событий: обнаружение, извещение и ответ
Промежуточное программное обеспечение
– Объектно-ориентированная парадигма в промежуточном программном обеспечении
– Брокеры объектных запросов
– Мониторы обработки транзакций
– Системы управления бизнес-процессами
– Современные инструментальные средства

Задачи обучения:

1. Объяснить и применить общепризнанные принципы построения высококачественных программных компонент.
2. Обсудить и выбрать архитектуру компонентной системы для заданного сценария.
3. Указать виды обработки событий, реализованные в одном или нескольких API.
4. Описать роль объектов в промежуточном программном обеспечении и отношения между компонентами.
5. Применить компонентно-ориентированные методы для создания нескольких программ в таких предметных областях, как параллелизм и транзакции, надежные коммуникационные службы, взаимодействие с базами данных, включая удаленные запросы и удаленное управление базами данных, безопасные коммуникации и доступ к данным.

SE10. Формальные методы [факультативный]

Темы:

Концепции формальных методов
Языки формальных спецификаций
Исполнимые и неисполнимые спецификации
Пред- и постусловия
Формальная верификация

Задачи обучения:

1. Применить методы формальной верификации к простым модулям программы.
2. Обсудить роль методов формальной верификации в контексте проверки соответствия и тестирования.
3. Описать потенциальные достоинства и недостатки использования языков формальных спецификаций.
4. Написать и оценить пред- и постусловия для нескольких ситуаций, от самых простых до более сложных.
5. Сформулировать спецификации простой программной системы на распространенном языке формальных спецификаций и продемонстрировать выгоды с точки зрения качества.

SE11. Надежность ПО [факультативный]

Темы:

Модели надежности ПО
Устойчивость и восстановление после сбоев
Классификация дефектов
Вероятностные методы анализа

Задачи обучения:

1. Продемонстрировать способность применять различные методы для оценки надежности программной системы.
2. Использовать избыточность и отказоустойчивость при разработке приложения среднего размера.
3. Объяснить проблемы, возникающие на пути достижения очень высокого уровня надежности.
4. Указать методы, которые позволяют реализовать программную архитектуру с требуемым уровнем надежности.

SE12. Разработка специализированных систем [факультативный]

Темы:

Системы реального времени
Системы "клиент-сервер"
Распределенные системы
Параллельные системы
Системы, основанные на Web
Сильно интегрированные системы

Задачи обучения:

1. Описать и обсудить различные специализированные системы.
2. Обсудить жизненный цикл и проблемы процесса разработки ПО для программных систем, разработанных в специализированном контексте.
3. Выбрать (с обоснованием выбора) методы, которые позволят эффективно разрабатывать и сопровождать специализированные программные системы.
4. Обсудить в заданном контексте и наборе связанных с ним профессиональных проблем, как разработчик ПО, занятый созданием специализированных систем, должен решать эти проблемы.
5. Обрисовать основные технические проблемы, связанные с разработкой специализированных систем.

Вычислительная математика и численные методы (CN)

CN1. Численный анализ [факультативный]
CN2. Исследование операций [факультативный]
CN3. Моделирование [факультативный]
CN4. Высокопроизводительные вычисления [факультативный]

Начиная с самых ранних дней информатики, вычислительная математика и численные методы (computational science and numerical methods) составляли значительную долю исследований. По мере того, как компьютеры становились способными решать все более сложные задачи, эта область – подобно всей дисциплине – приобретала все большее значение и важность. К концу двадцатого века научные вычисления утвердились в качестве самостоятельной дисциплины, имеющей тесные связи с информатикой, но, тем не менее, отличающейся от нее.

Несмотря на то, что курсы по численным методам и научным вычислениям чрезвычайно важны в качестве составных частей учебной программы по информатике, комитет CC2001 уверен, что ни один из разделов в данной области не представляет собой обязательные знания. По результатам исследований учебных программ и разговоров с преподавателями мы заключили, что нет единого мнения по поводу того, является ли этот материал важным для студентов, специализирующихся по информатике. Этот материал остается важной частью дисциплины, но не обязан входить во все учебные программы. Для тех студентов, которые выберут его изучение, область вычислений предлагает много ценных идей и методов, включая точность численного представления, анализ ошибок, численные методы, параллельные архитектуры и алгоритмы, моделирование и визуализацию научных данных. В то же время, студенты, выбирающие курсы по вычислениям, могут применить изучаемые методы в широком диапазоне прикладных областей, таких как:

- Молекулярная динамика
- Гидродинамика
- Небесная механика
- Экономическое прогнозирование
- Проблемы оптимизации
- Структурный анализ материалов
- Биоинформатика
- Вычислительная биология
- Геологическое моделирование
- Компьютерная томография

В большинстве учебных заведений каждому из разделов в данной области посвящен семестровый курс. Поэтому уровень детализации описаний тем и целей изучения отличается от используемого в документе для других областей, в которых на разделы отводится, как правило, меньше времени.

CN1. Численный анализ [факультативный]

Темы:

Арифметика с плавающей точкой
Ошибка, устойчивость, сходимость
Ряды Тейлора
Итеративные методы поиска корней уравнения (метод Ньютона)
Подбор кривых; приближение функций
Численное дифференцирование и интегрирование (правило Симпсона)
Явные и неявные методы
Дифференциальные уравнения (метод Эйлера)
Линейная алгебра
Конечные разности

Задачи обучения:

1. Сравнить различные методы численного анализа, представленные в этом разделе.
2. Определить понятия ошибки, устойчивости, машинной точности и погрешности приближенных вычислений.
3. Указать источники погрешности в приближенных вычислениях.
4. Спроектировать, запрограммировать, протестировать и отладить программы, реализующие численные методы.

CN2. Исследование операций [факультативный]

Темы:

Линейное программирование
– Целочисленное программирование
– Симплекс-метод
Вероятностное моделирование
Теория очередей
– Сети Петри
– Марковские цепи
Оптимизация

Анализ сетей и алгоритмы поиска путей
Предсказание и оценка
– Анализ решений
– Предсказание
– Управление рисками
– Эконометрика, микроэкономика
– Анализ чувствительности
Динамическое программирование
Примеры приложений
Программные средства

Задачи обучения:
1. Научиться применять основные методы исследования операций.
2. Описать несколько распространенных способов предсказания и оценки.
3. Спроектировать, запрограммировать, протестировать и отладить приложения, иллюстрирующие решение задач в области исследования операций.

CN3. Моделирование [факультативный]

Темы:
Случайные числа
– Генерация псевдослучайных чисел. Тестирование генераторов.
– Методы Монте-Карло
– Введение в функции распределения
Моделирование
– Дискретное моделирование
– Аналоговое моделирование
Верификация и проверка соответствия моделей
– Анализ входных данных
– Анализ выходных данных
Модели теории очередей
Примеры приложений

Задачи обучения:
1. Обсудить фундаментальные понятия компьютерного моделирования.
2. Оценить модели компьютерного моделирования.
3. Сравнить методы генерации случайных чисел.
4. Спроектировать, запрограммировать, протестировать и отладить программы, иллюстрирующие методы моделирования.

CN4. Высокопроизводительные вычисления [факультативный]

Темы:
Введение в высокопроизводительные вычисления
– История и важность вычислительных наук
– Обзор прикладных областей
– Обзор требуемых навыков
Высокопроизводительные вычисления
– Архитектуры процессоров
– Системы памяти для высокопроизводительных систем
– Устройства ввода/вывода
– Конвейерная обработка
– Параллельные языки и архитектуры
Аналитическая визуализация
– Представление результатов
– Форматы данных
– Инструментальные средства и пакеты визуализации данных
Примеры задач
– Модели океана и атмосферы
– Распространение сейсмической волны
– Системы N-тел (алгоритм Барнс-Хата)
– Химические реакции
– Агрегатные переходы
– Поток жидкости

Задачи обучения:

1. Привести пример предметных областей, в которых использование средств вычислительного моделирования способствует улучшению методов исследований.
2. Сравнить архитектуры для научных и параллельных вычислений, указать сильные и слабые стороны каждой.
3. Реализовать простые механизмы измерения производительности для высокопроизводительных систем.
4. Спроектировать, запрограммировать, протестировать и отладить программы, иллюстрирующие методы численного анализа, компьютерного моделирования и аналитической визуализации.

ПРИЛОЖЕНИЕ Б. ОПИСАНИЕ КУРСОВ.

Данное приложение к отчету CC2001 состоит из набора описаний курсов, предназначенных послужить моделью для учреждений, предлагающих обучение в области информатики. Хотя некоторые учебные учреждения, возможно, будут использовать данный набор с незначительными изменениями, образец курсов, приведенный здесь, намеренно сделан гибким и позволяет каждому учреждению подстроить его под собственные цели.

В большинстве случаев представленные ниже курсы уже присутствуют в учебной программе младших курсов, так как комиссия CC2001 ставила своей целью выделение и фиксирование успешного практического опыта, а не создание совершенно новых моделей. Поощряя развитие новых стратегий построения учебных планов и введение экспериментальных курсов, мы в то же время отдаем себе отчет, что разработка курсов занимает значительное время и требует практической и аудиторной оценки, которые не могут быть выполнены в рамках комиссии. Поэтому типовые курсы, приведенные в данном приложении, лучше всего рассматривать как некоторую отправную точку для экспериментирования. Несмотря на то, что каждый курс описан в достаточной степени подробности для непосредственного использования в учебном процессе, мы всячески приветствуем адаптацию и расширение этих курсов в динамичном процессе создания учебных планов во всех учебных учреждениях и на каждом отдельном факультете.

Основные концепции

Поскольку мы предполагаем, что приложения к отчету получат более широкое распространение, чем сам отчет, комитет считает важным включение в каждое приложение краткой сводки фундаментальных концепций, необходимых для понимания приведенных рекомендаций. Наиболее важные понятия описаны ниже в соответствующих разделах.

Полное обоснование учебного плана CC2001 приведено в полном отчете комитета. Однако мы предполагаем, что приложения получат широкое распространение и будут прочитаны многими людьми, у которых не будет времени для изучения полного отчета. Поэтому комитет считает важным включение в каждое приложение краткой сводки фундаментальных концепций, необходимых для понимания приведенных рекомендаций. Наиболее важными понятиями, необходимыми для понимания описаний курсов, являются следующие:

- *Совокупность знаний по информатике*. Курсы, описываемые в данном приложении, приведены в соответствие с приведенной выше классификацией знаний в области информатики. Эта классификация должна упорядочить общую **совокупность знаний** в области информатики. Совокупность знаний по информатике организована в виде трехуровневой иерархической структуры. На верхнем уровне иерархии находится **область**, представляющая собой отдельную часть дисциплины информатики. Каждая область обозначается двухбуквенной аббревиатурой, например, OS для *операционных систем* или PL для *языков программирования.* Области делятся на меньшие структуры, называемые **разделами**, которые представляют собой отдельные тематические модули внутри области. Каждый раздел обозначается численным суффиксом, добавляемым к имени области, например, OS3 обозначает раздел *параллелизма*. Каждый раздел, в свою очередь, состоит из набора **тем**, представляющих собой нижний уровень этой иерархии. Детальное изложение всех областей, разделов и тем приведено в Приложении А.

- *Обязательные и факультативные разделы.* В условиях расширения рамок компьютерных наук невозможно требовать от студента изучения всех тем, которые когда-либо рассматривались как фундаментальные. Поэтому комиссия решила определить минимальный набор **обязательных** курсов, включающий в себя только тот материал, который практически все преподаватели информатики признают необходимым для студентов, желающих получить диплом в области информатики. Так как обязательный материал по определению является минимальным, обязательные курсы сами по себе не могут составить полноценный учебный курс по специальности. Поэтому каждый учебный план должен включать дополнительные **факультативные** разделы (разделы по выбору), хотя данный отчет и не определяет, какие именно. Факультативные разделы, скорее всего, будут отличаться в зависимости от конкретного учебного учреждения, специализации и личных предпочтений каждого студента.

- *Вводные, основные и углубленные курсы*. В данном приложении курсы разделены на три категории в соответствии с тем уровнем, на котором они появляются в учебном плане. Курсы, определяемые как **вводные**, обычно читаются на первом году обучения в ВУЗе. Курсы, указанные как **основные**, обычно преподаются на втором или третьем году обучения и закладывают основу для дальнейшего обучения в данной области. Курсы, указанные как **углубленные**, обычно преподаются на последних годах обучения и концентрируются на темах, требующих значительной предварительной подготовки на более ранних курсах. Хотя данное разбиение достаточно ясно само по себе, важно не отождествлять уровень курса с понятиями *обязательный* и *факультативный*, которые относятся к разделам в совокупности знаний. Например, вводные и основные курсы определенно будут содержать преимущественно материалы обязательных разделов и все-таки полезно включать некоторые факультативные элементы уже в самые первые курсы. Аналогично, дополнительные курсы

будут содержать некоторую часть материалов обязательных курсов. Таким образом, эти разделения независимы и их не следует путать.

- *Часы.* Для того чтобы дать читателям представление о времени, необходимом для изучения отдельного раздела, документ CC2001 определяет стандартные метрики. Выбор такой метрики оказался сложной задачей, так как не существует общепринятой меры этой величины. Для согласования с ранними версиями документа, комиссия решила измерять время в **часах**, что соответствует аудиторным часам, необходимым для представления материала в традиционном формате, ориентированном на лекции. Заметим, что время, отводимое на раздел, не включает в себя время подготовки преподавателя и время, затрачиваемое студентами вне аудитории. В качестве рекомендации заметим, что объем внеаудиторных занятий должен примерно в три раза превосходить объем аудиторных. Так, раздел, требующий 3 часа должен обычно изучаться 12 часов (3 часа в аудитории и 9 часов самостоятельно). Также важно помнить, что для каждого раздела приводится *минимальное* время, и всегда будет полезным потратить больше времени, чем предлагаемый минимум.

Организация и формат описаний курсов

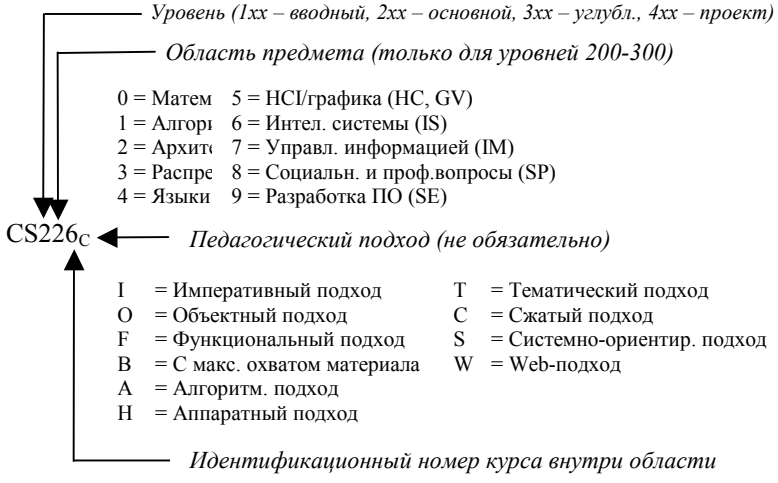
Как описано в предыдущем разделе, курсы, представленные в этом приложении, поделены на три уровня сложности: вводные, средние и дополнительные. Целью этого разбиения является создание естественных рамок, в которых развивается составление учебного плана. Например, глава 7, определяет шесть различных реализаций вводного курса. Глава 8 дает представление о четырех тематических подходах к составлению основных курсов вместе с набором смешанных стратегий, которые сочетают элементы каждого из этих подходов. Стратегии составления учебного плана и их связи показаны на рис. Б-1

В целом, должно быть возможным использовать любой подход к стилю изложения для вводных курсов и затем переходить к любому другому подходу для основных курсов, хотя подобное смешение стилей изложения может потребовать изучение некоторого переходного материала, чтобы гарантировать покрытие всех основных разделов. Необходимые стратегия и тактика для успешного перехода от одного стиля изложения к другому описаны в главах 6-8.

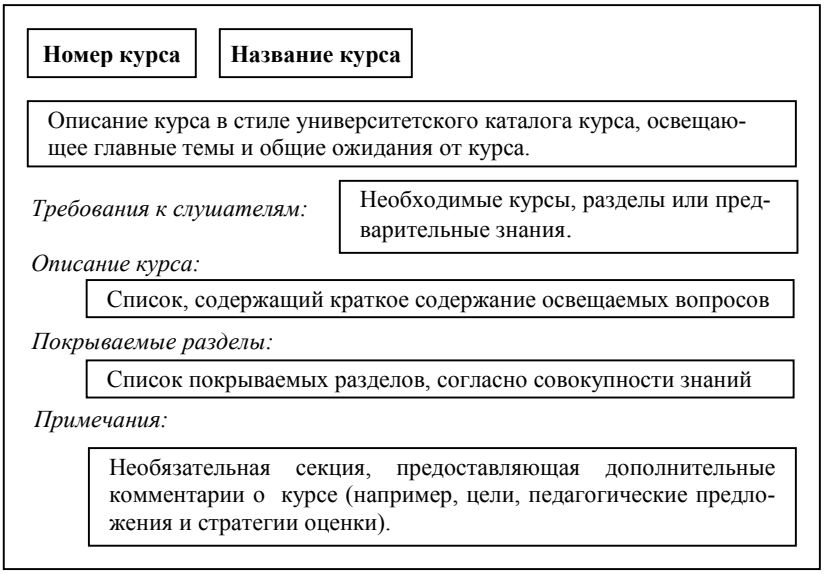
Рисунок Б-1. Уровни курсов и стратегии обучения

Названия индивидуальных педагогических подходов были выбраны так, чтобы они различались в начальных буквах. Это делает возможным именование курсов, которое одновременно кодирует уровень, область, и педагогический подход, как в примере, показанном на рисунке Б-2. В этом примере подстрочный индекс на конце названия $CS226_C$ обозначает, что данный курс основного уровня является частью сжатого подхода.

Рисунок Б-2. Схема нумерации курсов



Образец описания курса показан на рисунке Б-3, "компоненты курса". Изменяемые от описания к описанию части образца выделены рамкой.



Б.1. Модели для вводных курсов

В приведенных ниже описаниях курсов модели преподавания вводных курсов представлены в том порядке, в котором они описаны в главе 8.

Б.1.1. "Императивный" подход

Подход с ориентацией на императивное программирование (imperative-first approach) имеет две возможные реализации: в первой покрывается материал трех курсов ($CS101_I$, 102_I, 103_I), во второй используется более традиционная последовательность из двух курсов ($CS111_I$, $CS112_I$).

$CS101_I$. Основы программирования

Курс объясняет основные понятия процедурного программирования. Темы включают типы данных, управляющие структуры, функции, массивы, файлы и механизмы запуска, тестирования и отладки. Курс также содержит введение в исторический и социальный контекст компьютерных наук и обзор информатики как научной дисциплины.

Требования к слушателям: Никаких знаний в области программирования или информатики не требуется. Студенты должны иметь достаточный объем математических знаний для решения простых линейных уравнений и уметь пользоваться математической нотацией и формализмами.

Описание курса:

- Компьютерные приложения: обработка текстов, электронные таблицы, файлы и каталоги.
- Базовые конструкции программирования: синтаксис и семантика языка высокого уровня; переменные, типы, выражения и присваивания; простейший ввод/вывод; конструкции ветвления и итеративные конструкции; функции и передача параметров; структурная декомпозиция
- Алгоритмы и решение задач: стратегии решения задач, роль алгоритмов в решении задач, стратегии реализации алгоритмов, стратегии отладки, понятие алгоритма, свойства алгоритмов
- Базовые структуры данных: примитивные типы; массивы; структуры; строки и операции над строками
- Представление данных в памяти компьютера: биты, байты, слова; представление числовых данных и основания систем счисления; представление символьных данных
- Обзор операционных систем: роль и задачи операционных систем; простое управление файлами
- Введение в распределенные вычисления: основы и история сетей и Интернета; демонстрация и использование таких сетевых приложений, как электронная почта, telnet и FTP.
- Взаимодействие человека и машины: введение в вопросы проектирования
- Методология разработки ПО: основные понятия и принципы проектирования; структурная декомпозиция; стратегии тестирования и отладки; разработка тестовых наборов; среды разработки; инструменты тестирования и отладочные инструменты

- Социальный контекст программирования: история программирования и компьютеров; эволюция идей и аппаратуры; социальное воздействие компьютеров и Интернета; профессионализм, этический кодекс и ответственное поведение; авторские права, интеллектуальная собственность и нарушение авторских прав на программное обеспечение

Затрагиваемые разделы:

PF1	Основные конструкции программирования	10 часов (9 осн. + 1)
PF2	Алгоритмы и решение задач	3 осн. часа (из 6)
PF3	Основные структуры данных	2 осн. часа (из 14)
AR2	Представление данных в памяти компьютера	1 осн. час (из 3)
AR3	Организация машины на уровне ассемблера	2 осн. часа (из 9)
OS1	Обзор операционных систем	1 осн. час (из 2)
NC1	Введение в распределенные вычисления	1 осн. час (из 2)
PL3	Введение в трансляцию	1 осн. час (из 2)
PL4	Описание и типы данных	3 осн. часа
PL5	Механизмы абстракции	3 осн. часа
HC1	Основы взаимодействия человека и машины	1 осн. час (из 6)
GV1	Основы методов программирования графики	1 осн. час (из 2)
SP1	История информатики	1 осн. час
SP2	Социальные вопросы программирования	1 осн. час (из 3)
SP4	Профессиональная и этическая ответственность	1 осн. час (из 3)
SP6	Интеллектуальная собственность	1 осн. час (из 3)
SE1	Проектирование ПО	3 осн. часа (из 8)
SE3	Программные средства и окружения	2 осн. часа (из 3)
SE4	Процессы разработки ПО	1 осн. час (из 2)
	Темы по выбору	1 час

Примечания:

Этот курс представляет собой часть альтернативной реализации модели вводного курса с ориентацией на императивное программирование, в которой базовые понятия программирования покрываются за три семестра (а не за два, как в традиционном подходе). В терминах учебного плана это означает, что студенты должны приступать к освоению более сложных курсов после изучения последовательности $CS101_I$-102_I-103_I или двухсеместровой последовательности $CS111_I$-112_I, излагающей тот же материал в более концентрированной форме. Хотя преподавание основ программирования за два семестра в течение долгих лет было стандартом при обучении информатике, все большее число тем может со всеми основаниями считаться фундаментальными, создавая тем самым трудности в прочтении полного введения в информатику за один год. CC2001 ожидает, что трехсеместровая вводная последовательность курсов получит все более широкое распространение в ближайшие десять лет, и рекомендует кафедрам и факультетам экспериментировать с обеими этими моделями.

$CS102_I$. Объектно-ориентированная парадигма

Данный курс знакомит студентов, обладающих представлением о процеурной парадигме, с понятиями объектно-ориентированного программирования. Курс начинается с обзора управляющих структур и типов данных с акцентом на агрегатные типы данных и работу с массивами. Затем курс переходит к введению в объектно-ориентированную парадигму программирования, с особым упором на определении и использовании классов, а также на основах объектно-ориентированного проектирования. Другие темы курса включают в себя обзор принципов языков программирования, простой анализ алгоритмов, простейшие методы поиска и сортировки и введение в вопросы программной инженерии.

Требования к слушателям: $CS101_I$

Описание курса:

- Обзор управляющих структур, функций и примитивных типов данных
- Объектно-ориентированное программирование: объектно-ориентированное проектирование, инкапсуляция и скрытие информации; разделение интерфейса и реализации; классы, наследники и наследование; полиморфизм; иерархии классов
- Основные вычислительные алгоритмы: простейшие алгоритмы поиска и сортировки (линейный и дихотомический поиск, сортировка вставкой и выбором наименьшего элемента)
- Основы программирования событийно-управляемых систем
- Введение в компьютерную графику: использование простых графических API
- Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования

- Виртуальные машины: понятие виртуальной машины; иерархия виртуальных машин; промежуточные языки
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; стадии трансляции; машинно-зависимая и машинно-независимая части транслятора
- Введение в СУБД: история и причины развития баз данных, использование языка запросов базы данных
- Эволюция программ: сопровождение программ, характеристики сопровождаемого ПО, реинжиниринг, унаследованные системы, повторное использование ПО

Затрагиваемые разделы:

PF1	Основные конструкции программирования	3 осн. часа (из 9)
PF2	Алгоритмы и решение задач	6 осн. часов
PF3	Основные структуры данных	5 осн. часа (из 14)
PF5	Программирование событийно-управляемых систем	1 осн. час (из 4)
AL3	Основные вычислительные алгоритмы	3 осн. часа (из 12)
AR2	Представление данных в памяти компьютера	2 осн. часа (из 3)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL2	Виртуальные машины	1 осн. час
PL3	Введение в трансляцию	1 осн. час (из 2)
PL6	Объектно-ориентированное программирование	6 осн. часов (из 10)
HC1	Основы взаимодействия человека и машины	1 осн. час (из 6)
HC2	Построение простого графического интерфейса	2 осн. часа
IM2	СУБД	1 осн. час (из 3)
SE1	Проектирование ПО	1 осн. час (из 8)
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)
SE6	Проверка соответствия ПО	1 осн. час (из 3)
SE7	Эволюция программ	1 осн. час (из 3)
	Темы по выбору	1 час

Примечания:

Данный курс представляет собой второй семестр ориентированного на императивное программирование вводного курса, покрывающего базовые понятия программирования за три семестра, а не за два. Причины включения последовательности из трех курсов $CS101_I$-102_I-103_I как альтернативы к более традиционной двухсеместровой последовательности $CS111_I$-112_I подробно изложены в примечаниях к курсу $CS101_I$, а также обсуждаются в главе 7 данного отчета.

$CS103_I$. Алгоритмы и структуры данных.

Основываясь на познавательной базе курсов $CS101_I$-102_I, данный курс знакомит студентов с понятиями алгоритмов и структур данных. Темы курса включают в себя такие вопросы, как рекурсия, философия объектно-ориентированного программирования, базовые структуры данных (включая стеки, очереди, связные списки, хэш-таблицы, деревья и графы), основы анализа алгоритмов и введение в основы трансляции.

Требования к слушателям: $CS102_I$; желательно также знание дискретной математики в объеме курса CS105.

Описание курса:

- Обзор элементарных понятий программирования
- Базовые структуры данных: стеки, очереди, связные списки, хэш-таблицы, деревья, графы
- Объектно-ориентированное программирование: объектно-ориентированное проектирование, инкапсуляция и скрытие информации, классы, разделение интерфейса и реализации, иерархии классов, наследование, полиморфизм
- Основные вычислительные алгоритмы: алгоритмы сортировки со сложностью O(N logN), хэш-таблицы и алгоритмы исключения коллизий, двоичные деревья поиска, представления графов, обходы в глубину и в ширину
- Рекурсия: понятие рекурсии, рекурсивные математические функции, простые рекурсивные процедуры, стратегия "разделяй-и-властвуй", рекурсивный перебор с возвратами, реализация рекурсии
- Базовый анализ алгоритмов: асимптотический анализ наибольшей и средней сложности; установление различий между лучшим, средним и худшим случаями; O-большое и o-маленькое нотации; стандартные классы сложности; эмпирические измерения характеристик исполнения; затраты по объему памяти и времени; использование рекуррентных соотношений для анализа рекурсивных алгоритмов
- Обзор языков программирования: парадигмы программирования

- Программная инженерия: проверка соответствия ПО; основы тестирования, составление плана тестирования и генерация тестовых пакетов; тестирование объектно-ориентированных программ

Затрагиваемые разделы:

DS5	Графы и деревья	2 осн. часа (из 4)
PF3	Основные структуры данных	12 осн. часов (из 14)
PF4	Рекурсия	5 осн. часов
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL2	Алгоритмические стратегии	3 осн. часа (из 6)
AL3	Основные вычислительные алгоритмы	5 осн. часов (из 12)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL6	Объектно-ориентированное программирование	8 осн. часов (из 10)
SE6	Проверка соответсвия ПО	1 осн. час (из 3)

Примечания:

Данный курс представляет собой третий семестр ориентированного на императивное программирование вводного курса, покрывающего базовые понятия программирования за три семестра, а не за два. Причины включения последовательности из трех курсов $CS101_I$-102_I-103_I как альтернативы к более традиционной двухсеместровой последовательности $CS111_I$-112_I подробно изложены в примечаниях к курсу $CS101_I$, а также обсуждаются в главе 7 данного отчета.

$CS111_I$. Введение в программирование

Данный курс описывает базовые методы программирования, которые служат основой для дальнейшего, более углубленного изучения информатики. Значительное внимание уделяется эффективной практике разработки программного обеспечения с акцентом на такие аспекты как проектирование, декомпозиция, инкапсуляция, процедурная абстракция, тестирование и повторное использование программного обеспечения. Темы курса включают в себя стандартные конструкции программирования, стратегии решения задач, понятие алгоритма и базовых типов данных (строки, массивы, записи), а также введение в представление данных в памяти компьютера, введение в графику и сетевые технологии.

Требования к слушателям: Никаких знаний в области программирования или информатики не требуется. Студенты должны иметь достаточный объем математических знаний для решения простых линейных уравнений, а также для использования математической нотации и формализмов.

Описание курса:

- Основы: история программирования, обзор языков программирования и процесса компиляции
- Основные конструкции программирования: синтаксис и семантика высокоуровневого языка; переменные, типы, выражения и присваивание; простейший ввод/вывод; функции и передача параметров; структурная декомпозиция
- Алгоритмы и решение задач: стратегии решения задач, понятие алгоритма, свойства алгоритмов, стратегии реализации, последовательный и дихотомический поиск, квадратичные алгоритмы сортировки (выбор наименьшего и вставка)
- Основы теории вычислимости: легко разрешимые задачи и трудно разрешимые задачи, существование невычислимых функций
- Графика: использование графического программного интерфейса приложения
- Принципы инкапсуляции: инкапсуляция и скрытие информации, разделение интерфейса и реализации
- Основные структуры данных: примитивные типы; массивы; записи; строки и операции над строками; указатели и ссылки; статическое и стековое размещение, размещение в куче; динамическое управление памятью
- Представление данных в памяти компьютера: биты, байты и слова; двоичное представление целых; представление символьных данных; представление структур и массивов
- Организация машины на уровне ассемблера: организация фон-неймановской машины; выборка, дешифрация и выполнение команд; программирование на языке ассемблера для эмулируемой машины
- Методология разработки ПО: основные понятия и принципы проектирования, структурное проектирование, стратегии тестирования и отладки, разработка тестовых пакетов, среды программирования; средства тестирования и отладки

Затрагиваемые разделы:

PF1	Основные конструкции программирования	9 осн. Часов
PF2	Алгоритмы и решение задач	3 осн. часа (из 6)
PF3	Основные структуры данных	6 осн. часов (из 14)

AL3	Основные вычислительные алгоритмы	2 осн. часа (из 12)
AL5	Основы теории вычислимости	1 осн. час (из 6)
AR2	Представление данных в памяти компьютера	1 осн. час (из 3)
AR3	Организация машины на уровне ассемблера	2 осн. часа (из 9)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL4	Описание и типы данных	1 осн. час (из 3)
PL5	Механизмы абстракции	2 осн. часа (из 3)
PL6	Объектно-ориентированное программирование	3 осн. часа (из 10)
GV1	Основы методов программирования графики	2 осн. Часа
SP1	История информатики	1 осн. Час
SE1	Проектирование ПО	1 осн. час (из 8)
SE3	Программные средства и окружения	1 осн. час (из 3)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)
SE6	Проверка соответствия ПО	1 осн. час (из 3)
	Темы по выбору	1 час

Примечания:

Данный курс знакомит студентов с базовыми понятиями программирования, с акцентом на процедурную и императивную парадигмы. Большинство современных языков программирования, включая и большинство объектно-ориентированных, могут быть использованы для выполнения практических заданий по этому курсу. Действительно, вводные курсы, использующие объектно-ориентированные языки, обычно начинают с обсуждения их процедурных аспектов. От курса с ориентацией на объектно-ориентированное программирование $CS111_O$ данный курс отличается порядком тем и расстановкой приоритетов в них. В данном курсе обсуждение управляющих конструкций предшествует обсуждению классов, наследников и наследования; при объектно-ориентированном подходе порядок преподавания обратный.

Аналогично тому, как процедурные аспекты могут преподаваться на примере объектно-ориентированного языка программирования, так и некоторые базовые понятия объектно-ориентированного программирования могут быть найдены в традиционных императивных языках программирования. Упоминавшаяся в этом курсе тема "основы инкапсуляции" имеет смысл в обеих областях, но будет рассматриваться по-разному в зависимости от языка программирования. В любом случае, данный вопрос будет включать некоторые понятия раздела PL6, посвященного объектно-ориентированному программированию.

$CS112_I$. Абстракция данных

Данный курс продолжает введение в программирование, начатое в курсе $CS111_I$, фокусируясь на вопросах абстракции данных и объектно-ориентированного программирования. Темы курса включают в себя рекурсию, парадигмы программирования, принципы разработки языков программирования, виртуальные машины, объектно-ориентированное программирование, основные структуры данных и введение в теорию трансляции.

Требования к слушателям: $CS111_I$; также желательно знание дискретной математики в объеме курса CS105.

Описание курса:

- Обзор элементарного программирования
- Рекурсия: понятие рекурсии; рекурсивное задание математических функций (таких как факториал, числа Фибоначчи); простейшие рекурсивные процедуры (Ханойская башня, перестановки, фрактальные структуры); стратегии сортировки "разделяй-и-властвуй"; рекурсивный перебор с возвратами; реализация рекурсии
- Введение в теорию вычислительной сложности: асимптотический анализ предельной и средней сложности; нотация О-большое; стандартные классы сложности; эмпирические измерения характеристик выполнения программы
- Фундаментальные вычислительные алгоритмы: алгоритмы сортировки со сложностью O(N logN) (быстрая сортировка, иерархическая сортировка, сортировка слиянием); хеширование и стратегии исключения коллизий; двоичные деревья поиска
- Языки программирования: история языков программирования, краткий обзор парадигм программирования (процедурная, объектно-ориентированная, функциональная)
- Основные вопросы проектирования языков программирования: основные принципы разработки языков программирования; цели разработки; виды эквивалентности типов; модели данных; модели конструкций управления; механизмы абстракции
- Виртуальные машины: понятие виртуальной машины, иерархия виртуальных машин, промежуточные языки
- Объектно-ориентированное программирование: объектно-ориентированное проектирование; инкапсуляция и скрытие информации; разделение интерфейса и реализации; классы, наследники и наследование; полиморфизм; иерархии классов; коллекции и итераторы; основные шаблоны проектирования

- Основные структуры данных: связные структуры; стратегии реализации стеков, очередей, хэш-таблиц, графов и деревьев; стратегии выбора правильной структуры данных
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов, фазы трансляции (лексический анализ, синтаксический анализ, кодогенерация, оптимизация); машинно-зависимая и машинно-независимая части транслятора

Затрагиваемые разделы:

DS5	Графы и деревья	2 осн. часа (из 4)
PF3	Основные структуры данных	6 осн. часов (из 14)
PF4	Рекурсия	5 осн. часов
AL1	Основы анализа алгоритмов	2 осн. часа (из 4)
AL3	Основные вычислительные алгоритмы	4 осн. часа (из 12)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL2	Виртуальные машины	1 осн. час
PL3	Введение в теорию трансляции	2 осн. часа
PL4	Описание и типы данных	2 осн. часа (из 3)
PL5	Механизмы абстракции	1 осн. час (из 3)
PL6	Объектно-ориентированное программирование	7 осн. часов (из 10)
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
SE3	Программные средства и окружения	2 осн. часа (из 3)
	Темы по выбору	1 час

Примечания:

Как отмечено в описании требований к слушателям курса $CS111_I$, нет никакой гарантии, что студенты, посещающие данный курс, использовали объектно-ориентированный язык. В любом случае, при подходе с ориентацией на императивное программирование мы предполагаем, что вводный курс, даже если он использует объектно-ориентированный язык, концентрируется на императивных составляющих этого языка, а не на объектных механизмах. (Подход, начинающий с объектно-ориентированных понятий, изложен в последовательности курсов $CS111_O$/$CS112_O$). Одной из важных целей курса $CS112_I$ является знакомство студентов с объектно-ориентированной парадигмой в теории и на практике. Другими важными темами являются рекурсия, структуры данных и основные разделы из области Языков Программирования (PL), подходящие для изучения в данном курсе.

Б.1.2. "Объектно-ориентированный" подход

Как и подход с ориентацией на императивное программирование, данный подход (object-first approach) имеет две реализации – состоящую из трех ($CS101_O$-102_O-103_O) и из двух ($CS111_O$-112_O) курсов.

$CS101_O$. Введение в объектно-ориентированное программирование

Данный курс знакомит студентов с понятиями программирования в рамках объектной модели. Темы курса включают в себя простые типы данных, управляющие структуры, знакомство с массивами и структурами данных, алгоритмы для работы со строками, а также методы отладки и социальное значение программирования. Курс ставит своей целью усвоение студентами понятий, связанных с разработкой ПО, и развивает базовые навыки в программировании на примере языка, поддерживающего объектно-ориентированную парадигму.

Требования к слушателям: Никаких знаний в области программирования или информатики не требуется. Студенты должны иметь достаточный объем математических знаний для решения простых линейных уравнений, а также для использования математической нотации и формализмов.

Описание курса:

- Введение в историю информатики
- Этика и ответственность профессионалов-компьютерщиков
- Введение в компьютерные системы и среды
- Введение в объектно-ориентированную парадигму: абстракция, объекты, классы, методы, передача параметров, инкапсуляция, наследование, полиморфизм
- Основные конструкции программирования: основы синтаксиса и семантики высокоуровневого языка программирования; переменные, типы, выражения и присваивания; простейший ввод-вывод; условные и итеративные конструкции управления; структурная декомпозиция
- Базовые структуры данных: примитивные типы, массивы, структуры, строки и операции над строками
- Знакомство с языками программирования

- Алгоритмы и решение задач: стратегии поиска решения, роль алгоритма в поиске решения, стратегии реализации алгоритмов, стратегии отладки, понятие и свойства алгоритмов

Затрагиваемые разделы:

PF1	Основные конструкции программирования	9 осн. часов
PF2	Алгоритмы и решение задач	3 осн. часа (из 6)
PF3	Основные структуры данных	3 осн. часа (из 14)
AL3	Основные вычислительные алгоритмы	1 осн. час (из 12)
AL5	Основы теории вычислимости	1 осн. час (из 6)
AR2	Представление данных в памяти компьютера	2 осн. часа (из 3)
PL1	Обзор языков программирования	2 осн. часа
PL4	Описание и типы данных	2 осн. часа (из 3)
PL6	Объектно-ориентированное программирование	7 осн. часов (из 10)
PL8	Системы автоматического перевода языков	1 час
SP1	История информатики	1 осн. час
SP4	Профессиональная и этическая ответственность	1 осн. час (из 3)
SP5	Риски и ответственность компьютерных систем	1 осн. час (из 2)
SE3	Программные средства и окружения	1 осн. час (из 3)
SE6	Проверка соответствия ПО	1 осн. час (из 3)
	Темы по выбору	4 часа

Примечания:

Этот курс является частью альтернативной реализации объектно-ориентированного подхода к чтению вводного курса, который покрывает основные понятия программирования за три семестра, а не за два. В терминах учебного плана это означает, что студенты должны иметь возможность приступать к освоению более сложных курсов после изучения последовательности $CS101_O$-102_O-103_O или двухсеместровой последовательности $CS111_O$-112_O, которая излагает тот же материал в более концентрированной форме. Хотя преподавание основ программирования за два семестра в течение долгих лет было стандартом при обучении информатике, все большее число тем может со всеми основаниями считаться фундаментальными, создавая тем самым трудности в прочтении полного введения в информатику за один год. CC2001 ожидает, что трехсеместровая вводная последовательность курсов получит все более широкое распространение в ближайшие десять лет, и рекомендует кафедрам и факультетам экспериментировать с обеими этими моделями.

Отличие данного курса от реализации с ориентацией на императивное программирование заключается в том, что $CS101_O$-102_O-103_O с самого начало концентрирует внимание на объектах. В этом курсе обсуждение классов, наследников и наследования обычно предшествует обсуждению даже таких базовых понятий, как операторы ветвления и итеративные операторы.

$CS102_O$. Объекты и абстракция данных

Данный курс, вслед за $CS101_O$, продолжает знакомить студентов с методологией программирования с точки зрения объектно-ориентированного подхода. Излагая принципы проектирования объектов, данный курс также знакомит слушателей с основами человеко-машинного интерфейса, графики и социального значения программирования, с особым упором на вопросы компьютерной инженерии

Требования к слушателям: $CS101_O$

Описание курса:

- Обзор объектно-ориентированного программирования: объектно-ориентированная методология, объектно-ориентированное проектирование, программные средства
- Принципы объектно-ориентированного программирования: наследование, иерархии классов, полиморфизм, абстрактные классы и интерфейсы, классы-контейнеры/коллекции и итераторы
- Объектно-ориентированное проектирование: понятие шаблонов проектирования и использование программных интерфейсов приложения; программные средства моделирования, такие как диаграммы классов, CRC-диаграммы и UML use-cases
- Виртуальные машины: понятие виртуальной машины, иерархия виртуальных машин, промежуточные языки
- Основные вычислительные алгоритмы: поиск, сортировка, знакомство с рекурсивными алгоритмами
- Основные структуры данных: встроенные, создаваемые программистом и динамические структуры данных
- Программирование событийно-управляемых систем: методы обработки событий; распространение событий; обработка исключений
- Основы взаимодействия человека и компьютера: разработка и оценка эргономичных систем; основы хорошего дизайна; технические ограничения; основы тестирования эргономичности ПО

- Базовые методы программирования графики: иерархия графического ПО; использование графических программных интерфейсов приложения; простые модели цвета; однородные координаты; аффинные преобразования; преобразование изображений; отсечение
- Вопросы проектирования программного обеспечения: программные средства, процессы проектирования, требования, проектирование и тестирование, планирование повторного использования, риски и ответственность компьютерных систем

Затрагиваемые разделы:

PF3	Основные структуры данных	3 осн. часа (из 14)
PF4	Рекурсия	2 осн. часа (из 5)
PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
AL1	Базовый анализ алгоритмов	1 осн. час (из 4)
AL3	Основные вычислительные алгоритмы	2 осн. часа (из 12)
AR2	Представление данных в памяти компьютера	1 осн. час (из 3)
PL2	Виртуальные машины	1 осн. час
PL4	Описание и типы данных	1 осн. час (из 3)
PL5	Механизмы абстракции	3 осн. часа
PL6	Объектно-ориентированное программирование	7 осн. часов (из 10)
HC1	Основы взаимодействия человека и машины	1 осн. час (из 6)
GV1	Основы методов программирования графики	2 осн. часа
SE1	Проектирование ПО	3 осн. часа (из 8)
SE3	Программные средства и окружения	1 осн. часа (из 3)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)
SE6	Проверка соответствия ПО	1 осн. час (из 3)
SE7	Эволюция программ	1 осн. час (из 3)
	Темы по выбору	5 часов

Примечания:

Данный курс представляет собой второй семестр объектно-ориентированного вводного курса, покрывающего базовые понятия программирования за три семестра, а не за два. Причины включения последовательности из трех курсов $CS101_O$-102_O-103_O как альтернативы более традиционной двухсеместровой последовательности $CS111_O$-112_O подробно изложены в примечаниях к курсу $CS101_O$ и в главе 7 данного отчета.

$CS103_O$. Алгоритмы и структуры данных.

Основываясь на введении в объектно-ориентированное программирование, изложенном в курсах $CS101_O$-102_O, данный курс концентрируется на изучении алгоритмов, структур данных и проектирования программного обеспечения.

Требования к слушателям: $CS102_I$.

Описание курса:

- Обзор объектно-ориентированного проектирования
- Обзор основ разработки алгоритмов
- Обзор профессиональных и этических вопросов
- Алгоритмы и поиск решения задач: классические методы разработки алгоритмов; поиск решения задачи в объектно-ориентированной парадигме; применение методов разработки алгоритмов к проекту средних размеров с использованием формальным методам тестирования
- Основы анализа алгоритмов: асимптотический анализ максимальной и средней сложности; нахождение различий между лучшим, средним и худшим случаем; нотация О-большое; стандартные классы сложности; эмпирические измерения характеристик выполнения программы; компромиссы между алгоритмическим временем и алгоритмической памятью
- Рекурсия: понятие рекурсии, рекурсивные математические функции, простейшие рекурсивные процедуры, стратегия "разделяй-и-властвуй", рекурсивный перебор с возвратами, реализация рекурсии, рекурсия в применении к деревьям и графам
- Основные алгоритмы программирования: хэш-таблицы, двоичные деревья поиска, представление графов, обходы в ширину и в глубину, алгоритмы кратчайшего пути, транзитивное замыкание, минимальное остовное дерево, топологическая сортировка

- Основные структуры данных: указатели и ссылки; связные структуры; стратегии реализации списков, очередей и хэш-таблиц; стратегии реализации графов и деревьев; стратегии выбора подходящей структуры данных
- Программная инженерия: управление проектами, коллективное построение системы средних размеров с учетом эффективности алгоритмов

Затрагиваемые разделы:

PF2	Алгоритмы и решение задач	3 осн. часа (из 6)
PF3	Основные структуры данных	11 осн. часов (из 14)
PF4	Рекурсия	6 часов (5 осн. + 1)
AL1	Базовый анализ алгоритмов	3 осн. часа (из 4)
AL2	Алгоритмические стратегии	6 осн. часов
AL3	Основные вычислительные алгоритмы	5 осн. часов (из 12)
SE1	Проектирование ПО	1 осн. час (из 8)
SE8	Управление проектами	1 осн. час (из 3)
	Темы по выбору	4 часа

Примечания:

Данный курс представляет собой третий семестр объектно-ориентированного вводного курса, покрывающего базовые понятия программирования за три семестра, а не за два. Причины включения последовательности из трех курсов $CS101_O$-102_O-103_O как альтернативы к более традиционной двухсеместровой последовательности $CS111_O$-112_O подробно изложены в примечаниях к курсу $CS101_O$ и в главе 7 данного отчета.

$CS111_O$. Объектно-ориентированное программирование

Данный курс знакомит студентов с основными понятиями программирования, рассматривая их в рамках объектно-ориентированной модели. При изложении объектного проектирования, данный курс также затрагивает основы взаимодействия человека и компьютера, графику и социальное значение программирования, а также углубленно рассматривает проектирование программного обеспечения.

Требования к слушателям: Никаких знаний в области программирования или информатики не требуется. Студенты должны иметь достаточный объем математических знаний для решения простых линейных уравнений, а также для использования математической нотации и формализмов.

Описание курса:

- Основы: история программирования, обзор языков программирования и процесса компиляции
- Введение в объектно-ориентированное программирование: использование объектно-ориентированного языка, классы и объекты, синтаксис определений классов, методы, члены класса
- Простые данные: переменные, типы и выражения; присваивание
- Передача сообщений: простые методы, передача параметров
- Наследование классов
- Управляющие структуры: итеративные конструкции, условные операторы
- Алгоритмы: стратегии поиска решения задачи, понятие алгоритма, свойства алгоритмов, стратегии реализации алгоритмов
- Простейшие структуры данных: массивы, списки
- Классы-коллекции и правила итерации
- Использование программных интерфейсов приложений: библиотеки классов, пакеты для работы с графикой и создания GUI приложений
- Объектно-ориентированное проектирование: основные понятия и принципы проектирования, знакомство с шаблонами проектирования, объектно-ориентированный анализ и проектирование, проектирование с целью повторного использования
- Вопросы программной инженерии: программные средства, процессы разработки ПО, требования, проектирование и тестирование, риски и ответственность компьютерных систем

Затрагиваемые разделы:

PF1	Основные конструкции программирования	7 осн. часов (из 9)
PF2	Алгоритмы и решение задач	2 осн. часа (из 6)
PF3	Основные структуры данных	3 осн. часа (из 14)
PF4	Рекурсия	2 осн. часа (из 5)
AL3	Основные вычислительные алгоритмы	3 осн. часа (из 12)
AL5	Основы теории вычислимости	1 осн. час (из 6)
PL4	Описание и типы данных	2 осн. часа (из 3)

PL5	Механизмы абстракции	1 осн. час (из 3)
PL6	Объектно-ориентированное программирование	8 осн. часов (из 10)
GV1	Основы методов программирования графики	2 осн. часа
SP1	История информатики	1 осн. час
SP5	Риски и ответственность компьютерных систем	1 осн. час (из 2)
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	1 осн. час (из 5)
SE3	Программные средства и окружения	2 осн. часа (из 3)
	Темы по выбору	2 часа

Примечания:

Данный курс знакомит студентов с базовыми понятиями программирования, с самого начала используя объектно-ориентированную парадигму. От аналогичного курса $CS111_I$ данный курс отличает то, что обсуждение классов, наследников и наследования предшествует знакомству даже с такими основными понятиями, как условные и итеративные операторы.

Чтобы проиллюстрировать степень внимания, уделяемого в курсе объектам, обратимся к широко распространенному примеру приложения, простому числовому калькулятору, который подходит как пример приложения для вводного курса как в "императивном" подходе к обучению, так и в "объектно-ориентированном". В рамках парадигмы императивного программирования такая программа будет организована как цикл, запрашивающий команды пользователя и использующий условную передачу управления, например, с помощью оператора **`switch`** в семействе языков C, для выполнения действий, соответсвующих каждой операции. Объектно-ориентированная реализация, скорее всего, не будет иметь явных циклов и условных предложений. Вместо этого, кнопки калькулятора будут частью иерархии объектов. Кнопки разрядов, например, могут быть экземплярами класса цифровых кнопок, общим поведением которого будет добавление соответствующего разряда к концу отображаемого значения; отдельные экземпляры цифровых кнопок могут отличаться только значением переменной, хранящей цифру, которой соответствует кнопка. Кнопка с плюсом будет реализовывать метод operate одним образом, кнопка с минусом – другим. Использование объектно-ориентированной парадигмы позволяет студентам решать задачи, подобные созданию иерархии объектов для калькулятора, гораздо раньше, так как количество и сложность необходимых изучаемых структур управления значительно меньше.

Большинство курсов с упором на объектно-ориентированное программирование будут использовать среду программирования, предоставляющую широкие возможности по использованию программных интерфейсов приложений, API. Эти API могут неоценимо помочь студентам, так как они позволяют создавать более интересные программы уже на ранних этапах обучения, увеличивая мотивацию студентов. В то же время, объемы и возможности большинства API пакетов из-за огромного набора классов и методов могут быть пугающими для многих студентов. Чтобы уменьшить эффект подобной понятийной перегрузки, преподавательский состав может использовать собственные аналогичные библиотеки с ограниченной функциональностью.

$CS112_O$. Объектно-ориентированное проектирование и методология

Данный курс, как и курс $CS111_O$, продолжает знакомить студентов с объектно-ориентированным программированием, уделяя основное внимание алгоритмам, структурам данных, проектированию ПО и социальным вопросам программирования.

Требования к слушателям: $CS111_O$.

Описание курса:

- Обзор объектно-ориентированного проектирования и программирования, включая обзор программных средств
- Обзор простейших приемов разработки алгоритмов с учетом этической и социальной ответственности (например, необходимость тестирования)
- Классические методы разработки и реализации алгоритмов, а также их место в проектировании объектно-ориентированных систем
- Абстракция и инкапсуляция путем использования классических структур данных: знакомство (т.е. использование существующих, а не реализация) с классическими структурами данных (списков, стеков и очередей), а также их связь с разработкой алгоритмов
- Введение в основы анализа алгоритмов
- Применение методов разработки алгоритмов в проекте средних размеров с использованием формальных методов тестирования
- Рекурсия: рекурсия как метод проектирования, реализация рекурсии и ее связь с итерацией, знакомство с деревьями и графами

- Введение в распределенные алгоритмы: консенсус и выборы; отказоустойчивость
- Программная инженерия: создание проекта средних размеров в команде с учетом алгоритмической эффективности реализации

Затрагиваемые разделы:

PF1	Основные конструкции программирования	2 осн. часа (из 9)
PF2	Алгоритмы и решение задач	2 осн. часа (из 6)
PF3	Основные структуры данных	8 осн. часов (из 14)
PF4	Рекурсия	3 осн. часа (из 5)
PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL2	Алгоритмические стратегии	2 осн. часа (из 6)
AL3	Основные вычислительные алгоритмы	3 осн. часа (из 12)
PL1	Обзор языков программирования	2 осн. часа
PL2	Виртуальные машины	1 осн. час
PL4	Описание и типы данных	1 осн. час (из 3)
PL5	Механизмы абстракции	2 осн. часа (из 3)
PL6	Объектно-ориентированное программирование	4 осн. часа (из 10)
HC1	Основы взаимодействия человека и машины	1 осн. час (из 6)
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	1 осн. час (из 5)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)
SE6	Проверка соответствия ПО	1 осн. час (из 3)

Примечания:

Данный курс основывается на знаниях, полученных студентами в курсе $CS111_O$, и завершает годичный вводный обзор программирования. Так как курс $CS111_O$ уделяет больше внимания механике объектно-ориентированного программирования, чем обычный курс с ориентацией на императивное программирование, в данном курсе можно уделить больше времени вопросам проектирования программного обеспечения и программной инженерии, не нанося серьезного ущерба традиционному материалу, связанному со структурами данных и алгоритмами.

Б.1.3. "Функциональный" подход

Подход, ориентированный на функциональное программирование, существует только в двухсеместровой форме. Если данный подход будет активно использоваться, возможно, появится необходимость рассмотреть и трехсеместровый вариант.

$CS111_F$. Введение в функциональное программирование

Данный курс знакомит студентов с основными понятиями программирования в рамках функционального языка, подчеркивая значимость алгоритмических стратегий, а не деталей синтаксиса языка.

Требования к слушателям: отсутствуют.

Описание курса:

- Обзор истории программирования
- Процедурная абстракция: простые функции, параметры и результаты, композиция, условные выражения
- Рекурсия: понятие рекурсии, рекурсивные математические функции, простые рекурсивные процедуры
- Абстракция данных: списковая структура, иерархические данные, символьные данные, важность абстракции данных
- Алгоритмы и поиск решений: стратегии поиска решений задач, роль алгоритмов в процессе решения задач, стратегии реализации алгоритмов, стратегии отладки, понятие и свойства алгоритмов
- Алгоритмические стратегии: методы решения в лоб; жадные алгоритмы; "разделяй-и-властвуй"; возвраты; алгоритмы численных приближений
- Основы теории вычислимости: легко и трудно разрешимые задачи; существование невычислимых функций
- Основы теории вычислительной сложности: асимптотический анализ наибольшей и средней сложности, О-нотация, стандартные классы сложности, эмпирические измерения характеристик выполнения программы
- Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования; роль трансляции в процессе программирования
- Стратегии вычисления: представление состояния, потоки, ленивое вычисление, недетерминизм, структура интерпретатора

- Представление данных в памяти компьютера: биты, байты и слова; численные данные и основания систем счислений; знаковые представления и представления в дополнительном коде; представление нечисловых данных

Затрагиваемые разделы:

DS5	Графы и деревья	3 осн. часа (из 4)
PF1	Основные конструкции программирования	3 осн. часа (из 9)
PF2	Алгоритмы и решение задач	2 осн. часа (из 6)
PF3	Основные структуры данных	6 осн. часов (из 14)
PF4	Рекурсия	5 осн. часов
OS3	Параллелизм	2 осн. часа (из 6)
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL2	Алгоритмические стратегии	2 осн. часа (из 6)
AL3	Основные вычислительные алгоритмы	4 осн. часа (из 12)
AL5	Основы теории вычислимости	1 осн. час (из 6)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL4	Описание и типы данных	1 осн. час (из 3)
PL5	Механизмы абстракции	1 осн. час (из 3)
PL7	Функциональное программирование	4 осн. часа (из 7)
SP1	История информатики	1 осн. час
SP5	Риски и ответственность компьютерных систем	1 осн. час (из 2)
SE1	Проектирование ПО	1 осн. час (из 8)
SE3	Программные средства и окружения	1 осн. час (из 3)

$CS112_F$. Объекты и алгоритмы

Данный курс углубляет знания, полученные в курсе $CS111_F$, расширяя их понятиями объектно-ориентированного программирования и проектирования.

Требования к слушателям: $CS111_F$.

Описание курса:

- Основные структуры программирования: основы синтаксиса и семантики высокоуровневого языка; переменные, типы, выражения и присваивание; простейший ввод-вывод; конструкции ветвления и итеративные конструкции; функции и передача параметров, структурная декомпозиция
- Объектно-ориентированное программирование: объектно-ориентированное проектирование; инкапсуляция и сокрытие данных; разделение реализации и интерфейса; классы, наследники и наследование, полиморфизм, иерархии классов, классы-коллекции и правила итерации, основные шаблоны проектирования
- Основные структуры данных: примитивные типы, массивы, структуры, строки и операции со строками, указатели и ссылки, связные структуры, стратегии выбора правильной структуры данных
- Программирование событийно-управляемых систем и параллелизма: методы обработки событий; распространение событий; управление параллельным исполнением при обработке событий; обработка исключений
- Использование программных интерфейсов приложений: программы просмотра классов и другие подобные инструменты; программирование на примерах; отладка программы, использующей программный интерфейс приложения
- Алгоритмические стратегии: алгоритмы решения задачи в лоб, жадные алгоритмы, методы "разделяй-и-властвуй", возвраты, эвристики
- Основные вычислительные алгоритмы: простые численные алгоритмы; последовательные и дихотомический поиск; алгоритмы сортировки
- Виртуальные машины: понятие виртуальной машины; классификация виртуальных машин; промежуточные языки; вопросы безопасности, возникающие при запуске кода на удаленной машине
- Основные методы программирования графики: классификация графического ПО, использование графических API
- Методология разработки ПО: основные понятия и принципы; структурное проектирование, стратегии тестирования и отладки, создание тестовых пакетов, среды программирования, программные средства тестирования и отладки

Затрагиваемые разделы:

PF1	Основные конструкции программирования	6 осн. часов (из 9)
PF2	Алгоритмы и решение задач	1 осн. час (из 6)
PF3	Основные структуры данных	5 осн. часов (из 14)
PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
AL2	Алгоритмические стратегии	2 осн. часа (из 6)

AL3	Основные вычислительные алгоритмы	2 осн. часа (из 12)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL2	Виртуальные машины	1 осн. час
PL4	Описание и типы данных	2 осн. часа (из 3)
PL5	Механизмы абстракции	2 осн. часа (из 3)
PL6	Объектно-ориентированное программирование	8 осн. часов (из 10)
SE1	Проектирование ПО	3 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
SE3	Программные средства и окружения	1 осн. час (из 3)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)
SE6	Проверка соответствия ПО	1 осн. час (из 3)

Б.1.4. Обучение с максимальным охватом материала

Как отмечено в главе 8, мы предлагаем две реализации подхода с максимальным охватом материала (breadth-first approach). Первая реализация включает обзорный курс $CS100_B$ перед более традиционной последовательностью курсов. Вторая реализация предлагает провести вводный курс в виде трехсеместровой последовательности ($CS101_B$-102_B-103_B), обеспечив, тем самым, время для дополнительных тем.

$CS100_B$. Обзор информатики

Данный курс предлагает обзор информатики, дающий студентам возможность понять и оценить множество различных аспектов информатики. Темы данного курса включают в себя: дискретную математику, знакомство с языками программирования, алгоритмическое решение задач, анализ алгоритмической сложности, основные понятия архитектуры ЭВМ, операционные системы и сети, графику и социальные вопросы программирования. Никаких предварительных знаний по информатике данный курс не предполагает и не требует. Курс предназначен как для студентов, собирающихся специализироваться в области информатики, так и для студентов других специальностей.

Требования к слушателям: отсутствуют.

Описание курса:

- Математическое введение: множества, функции, логика, доказательства
- Алгоритмы: определение, создание и реализация; знакомство с классическими алгоритмами (сортировка, поиск и сопоставление с образцом)
- Анализ алгоритмов: эффективность, асимптотический анализ, вычислительная сложность, О-нотация, полиномиальное и экспоненциальное возрастание, вычислимость
- Аппаратные реализации алгоритмов: представление данных, модель вычислений фон Неймана, цикл выборка-декодирование-исполнение, основы архитектуры ЭВМ
- Основы программирования: обзор основ программирования и принципы объектно-ориентированного проектирования, краткое знакомство с языком программирования, поддерживающим объектно-ориентированную парадигму
- Операционные системы и виртуальные машины: эволюция операционных систем, задачи операционной системы, основные компоненты операционной системы
- Сети и компьютерная графика: краткое знакомство с некоторыми базовыми понятиями сетевых технологий и компьютерной графики
- Социальные и профессиональные вопросы: социальные вопросы программирования, ответственности профессионалов в области информационных технологий

Затрагиваемые разделы:

DS1	Функции, отношения и множества	2 осн. часа (из 6)
DS2	Основы логики	2 осн. часа (из 10)
PF1	Основные конструкции программирования	5 осн. часов (из 9)
PF2	Алгоритмы и решение задач	3 осн. часа (из 6)
AL1	Базовый анализ алгоритмов	4 осн. часа
AL3	Основные вычислительные алгоритмы	4 осн. часа (из 12)
AR6	Функциональная организация устройств	4 осн. часа (из 7)
OS1	Обзор операционных систем	2 осн. часа
OS2	Основы операционных систем	1 осн. час (из 2)
NC1	Введение в распределенные вычисления	2 осн. часа
NC2	Телекоммуникации и сети	1 осн. час (из 7)
PL6	Объектно-ориентированное программирование	4 осн. часа (из 10)
GV1	Основы методов программирования графики	2 осн. часа
GV2	Графические системы	1 осн. час
SP2	Социальные вопросы программирования	3 осн. часа

Примечания:

Конечно, невозможно охватить все разнообразие тем по информатике в одном курсе. Поэтому точный список тем и их порядок может меняться в зависимости от интересов и знаний преподавателя. Как минимум, курс, ориентированный на максимальный охват материала, должен начинаться с основательного изложения базовых алгоритмов, основных понятий устройства компьютера, с описания абстракции виртуальных окружений, создаваемых программными средствами, краткого введения в программирование и проектирование ПО, а также обсуждения социальных, профессиональных и этических вопросов, которые существуют в области информатики. Помимо этих тем, каждый преподаватель может выбирать любые дополнительные темы (особенно рекомендуются темы, освещающие современные приложения программирования). Примерный план включает около шести часов, посвященных материалу о сетевых технологиях и компьютерной графике, двум важным и быстро развивающимся областям. Впрочем, было бы уместным расширить и другие темы, такие как базы данных, искусственный интеллект и распределенные системы.

Существуют два важных соображения, которые необходимо учитывать при составлении курса, ориентированного на максимальный охват материала. Первое соображение заключается в том, что дискретная математика должна рассматриваться как неотъемлемая часть курса. При таком подходе студенты смогут лучше понять и оценить важность дискретной математики в информатике. Например, булевская логика может быть рассмотрена в процессе обсуждения операторов языка, алгоритмы счета могут рассматриваться при обсуждении эффективности итеративных алгоритмов, а рекуррентные соотношения вполне естественно изучать применительно к вопросу эффективности рекурсивных алгоритмов. Цель состоит в том, чтобы студенты получили представление о математических понятиях в контексте их использования при решении важных вычислительных задач.

Второе соображение состоит в том, что многие малосвязанные темы, обычно встречаемые в программе курса, ориентированного на максимальный охват материала, должны быть интегрированы в единое целое. Студенты не должны воспринимать курс как набор интересных, но не связанных между собой тем, вроде "во вторник будет устройство компьютера". Вместо этого, необходимо развить в студентах понимание сложных взаимосвязей между основными областями информатики. Эта цель может быть достигнута путем демонстрации того, как каждая из тем курса использует идеи из уже пройденного материала и на их основе получает новые и более мощные абстракции. Такой "спиральный" подход является важной составляющей успеха курса.

$CS101_B$. Введение в информатику

Данный курс предлагает студентам развернутое введение в информатику, связывающее воедино идеи программирования, дискретной математики, основ архитектуры ЭВМ, теории алгоритмов и теории вычислимости.

Требования к слушателям: не требуется никакого опыта в области программирования. Студенты должны иметь достаточный объем математических знаний для решения простых линейных уравнений, а также для использования математической нотации и формализмов.

Описание курса:

- Дискретная математика: функции, отношения и множества; основы логики; методы доказательства; основы вычислений; дискретная вероятность
- Основные конструкции программирования: основы синтаксиса и семантики высокоуровневого языка программирования; переменные, выражения и присваивание; простейший ввод-вывод; операторы ветвления и итеративные операторы; функции и передача параметров; структурная декомпозиция
- Алгоритмы и поиск решений задач: стратегии поиска решений задач; роль алгоритмов в процессе поиска решений задач; понятие и свойства алгоритмов
- Основные структуры данных: примитивные типы, массивы, строки и операции над строками
- Рекурсия: понятие рекурсии, рекурсивные математические функции, стратегии поиска "разделяй-и-властвуй"
- Основы анализа алгоритмов: О-нотация, стандартные классы сложности
- Основные вычислительные алгоритмы: конечные автоматы; машины Тьюринга; легко и трудно разрешимые задачи; невычислимые функции; проблема останова; следствия невычислимости
- Обзор языков программирования: история языков программирования
- Цифровая логика и цифровые системы: обзор и история архитектуры ЭВМ, базовые аппаратные блоки, логические выражения
- История программирования
- Знакомство с социальным значением программирования

Затрагиваемые разделы:

DS1	Функции, отношения и множества	4 осн. часа (из 6)
DS2	Основы логики	5 осн. часов (из 10)
DS3	Методы доказательств	4 осн. часа (из 12)
DS4	Основы вычислений	3 осн. часа (из 5)
DS6	Дискретная вероятность	4 осн. часа (из 6)
PF1	Основные конструкции программирования	3 осн. часа (из 9)
PF2	Алгоритмы и решение задач	2 осн. часа (из 6)
PF3	Основные структуры данных	2 осн. часа (из 14)
PF4	Рекурсия	2 осн. часа (из 5)
AL1	Базовый анализ алгоритмов	1 осн. час (из 4)
AL3	Основные вычислительные алгоритмы	2 осн. часа (из 12)
AL5	Основы теории вычислимости	1 осн. час (из 6)
AR1	Цифровая логика и цифровые системы	2 осн. часа (из 6)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL3	Введение в трансляцию	1 осн. час (из 2)
PL4	Описание и типы данных	3 осн. часа
SP1	История информатики	1 осн. час
SP2	Социальные вопросы программирования	1 осн. час (из 3)

Примечания:

Данный курс является первым курсом из трехсеместровой последовательности ($CS101_B$-102_B-103_B), дающей широкое интегрированное введение в информатику в соответствии с рекомендациями отчетов "Компьютерные науки как дисциплина" от 1989 года [Denning89] и CC1991 [Tucker91]. Каждый из трех курсов последовательности включает в себя как теорию, так и практику программирования. Кроме того, каждый курс последовательности включает в себя набор дополнительных тем.

Как было отмечено в главе 7, подход с максимальным охватом материала не снискал тех лавров, которые ему прочили его пропагандисты. Однако мы верим, что относительный провал такого подхода частично может быть оправдан тем, что была произведена попытка изложить все разнообразие информатики за слишком малое количество часов. Учитывая постоянный рост объема дисциплин программирования, преподаватели попросту не имеют возможности достаточно подробно осветить все понятия в рамках двух семестров. В результате большинство курсов, ориентирующихся на максимальный охват материала, сегодня служат не более, чем вводными курсами к более традиционным схемам преподавания. Такая модель, у которой есть несколько успешных реализаций, представлена в описании курса $CS100_B$.

В течение нескольких последних лет рамки двух семестров стали слишком тесными даже для чисто программистского материала. В результате, многие учебные учреждения переходят к трехсеместровой последовательности вводных курсов. Мы рекомендуем такой подход в разделе 7.3.3 данного документа, а также предлагаем образец реализации в курсе $CS100_B$. Интересно отметить, что переход к трехсеместровой системе может сделать подход с максимально широким охватом материала более жизнеспособным.

Материал, представленный в курсах $CS101_B$-102_B-103_B, достаточно похож на предлагаемый в традиционных двухсеместровых последовательностях и курсе по дискретным структурам CS115. Разница заключается в организации материала. При подходе с максимальным охватом материала, математика распределена по всем трем семестрам и непосредственно привязана к темам, в которых она используется. Таким образом, студенты могут лучше оценить связь между теорией и практикой.

Главной опасностью для всех подходов с максимальным охватом материала является то, что студенты обычно больше интересуются материалом, связанным непосредственно с программированием, а не теорией. Чтобы удовлетворить это стремление, мы постарались включить в первый курс последовательности больше программирования, чем сейчас принято. С точки зрения преподаваемых разделов, можно сказать, что треть материала $CS101_B$ посвящена непосредственно программированию, а большинство остальных тем могут быть изложены с акцентом на практическое применение.

Мы осознаем, что наша реализация подхода с максимальным охватом материала не была протестирована, и поэтому ее может постичь участь всех предшественниц. Тем не менее, мы надеемся, что расширение временных рамок до трех семестров поможет справиться с недостатками предыдущих реализаций. В конце концов, существуют вполне успешные схемы, включающие в себя один предварительный курс, ориентированный на максимальный охват материала, и два курса, реализующие другой подход. Преимуществом же интегрированного устройства курсов в рамках единого подхода с максимальным охватом материала является возможность предоставить студентам больше программистской практики на ранних этапах с углублением в теоретические вопросы в последующих курсах.

$CS102_B$. Алгоритмы и методы программирования

Данный курс знакомит студентов с программированием на основе полученных ими ранее знаний.

Требования к слушателям: $CS101_B$.

Описание курса:

- Дискретная математика: основы логики, методы доказательства
- Алгоритмы и поиск решений задач: стратегии реализации алгоритмов, стратегии отладки
- Основные конструкции программирования: модели описания данных, сборка мусора, механизмы абстракции, модули
- Основные структуры данных: массивы; структуры; строки и работа со строками; представление данных в памяти; статическое, стековое выделение памяти и выделение памяти в куче; динамическое управление типом хранения; указатели и ссылки
- Объектно-ориентированное программирование: инкапсуляция, и скрытие информации; разделение интерфейса и реализации; классы, наследники и наследование; полиморфизм; иерархии классов
- Основные вычислительные алгоритмы: простые численные алгоритмы, хэш-таблицы
- Обзор языков программирования: краткий обзор парадигм программирования
- Виртуальные машины: понятие виртуальной машины, иерархии виртуальных машин, промежуточные языки
- Машинное представление данных: биты, байты и слова; представление численных данных и основания систем счисления; представления с фиксированной и с плавающей точкой; знаковые представления и представления в дополнительном коде; представление нечисловых данных; представление массивов и структур
- Организация машины на уровне ассемблера: основы организации фон неймановской машины; блок управления; выборка, декодирование и выполнение инструкций
- Введение в распределенное программирование: причины возникновения и история сетевого программирования и Интернета; сетевые архитектуры
- Построение простого графического интерфейса пользователя: основы графических интерфейсов пользователя; инструментарий для создания графических интерфейсов пользователя
- Разработка ПО: проектирование программного обеспечения; программные средства и окружения; требования и требования; проверка соответствия ПО; стратегии тестирования и отладки

Затрагиваемые разделы:

DS2	Основы логики	5 осн. часов (из 10)
DS3	Методы доказательств	2 осн. часа (из 12)
PF1	Основные конструкции программирования	5 осн. часов (из 9)
PF2	Алгоритмы и решение задач	1 осн. час (из 6)
PF3	Основные структуры данных	3 осн. часа (из 14)
PF4	Рекурсия	1 осн. час (из 5)
PF5	Программирование событийно-управляемых систем	1 осн. час (из 4)
AL3	Основные вычислительные алгоритмы	1 осн. час (из 12)
AR1	Цифровая логика и цифровые системы	1 осн. час (из 6)
AR2	Представление данных в памяти компьютера	1 осн. час (из 3)
AR3	Организация машины на уровне ассемблера	1 осн. час (из 9)
NC1	Введение в распределенные вычисления	1 осн. час (из 2)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL2	Виртуальные машины	1 осн. час
PL4	Описание и типы данных	2 осн. часа (из 3)
PL5	Механизмы абстракции	2 осн. часа (из 3)
PL6	Объектно-ориентированное программирование	3 осн. часа (из 10)
HC1	Основы взаимодействия человека и машины	1 осн. час (из 6)
HC2	Построение простого графического интерфейса	2 осн. часа
SE1	Проектирование ПО	2 осн. часа (из 8)
SE3	Программные средства и окружения	1 осн. час (из 3)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)
SE6	Проверка соответствия ПО	1 осн. час (из 3)

Примечания:

Данный курс является вторым курсом в трехсеместровой последовательности ($CS101_B$-102_B-103_B), пытающейся дать общее интегрированное введение в информатику. Причины создания данной последовательности, а также предложения для данной реализации приведены в примечаниях к курсу $CS101_B$.

$CS103_B$. Принципы объектно-ориентированного проектирования

Данный курс предлагает студентам возможность расширить свои знания в области объектно-ориентированного программирования и принципов объектно-ориентированного проектирования.

Требования к слушателям: $CS102_B$.

Описание курса:

- Дискретная математика: функции, отношения и множества; методы доказательства; решение рекуррентных соотношений; математические свойства графов и деревьев; дискретная вероятность
- Основные конструкции программирования: итераторы и модели итераторов; рекурсивные структуры данных
- Основные структуры данных: стратегии реализации для стеков, очередей, хэш-таблиц, графов и деревьев; стратегии выбора правильной структуры данных
- Использование программных интерфейсов приложений: программирование с использованием API; отладка программ, использующих API; введение в программирование компонент
- Анализ алгоритмов: асимптотический анализ наибольшей и средней сложности; установление различий между лучшим, средним и худшим случаями; O-нотация; эмпирические измерения характеристик исполнения; затраты по объему памяти и времени; использование рекуррентных соотношений для анализа рекурсивных алгоритмов
- Основные алгоритмы программирования: двоичные деревья поиска; представление графов; обходы в глубину и в ширину; алгоритмы кратчайшего пути; транзитивное замыкание; минимальное остовное дерево; топологическая сортировка
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; фазы трансляции; машинно-зависимая и машинно-независимая части транслятора; стратегии синтаксического разбора
- Объектно-ориентированное программирование: объектно-ориентированное проектирование; инкапсуляция и скрытие информации; разделение интерфейса и реализации; классы-коллекции и правила итерации; внутреннее представление объектов и таблицы методов
- Обзор операционных систем: роль и задачи операционных систем; история развития операционных систем; функциональные возможности типичной операционной системы
- Основные вопросы интеллектуальных систем: история искусственного интеллекта; философские вопросы; основные определения; моделирование мира; роль эвристик
- Разработка ПО: объектно-ориентированный анализ и проектирование; проектирование для повторного использования; шаблоны проектирования; программные окружения; программные средства тестирования

Затрагиваемые разделы:

DS1	Функции, отношения и множества	2 осн. часа (из 6)
DS3	Методы доказательств	3 осн. часа (из 12)
DS4	Основы вычислений	2 осн. часа (из 5)
DS5	Графы и деревья	2 осн. часа (из 4)
DS6	Дискретная вероятность	2 осн. часа (из 6)
PF1	Основные конструкции программирования	1 осн. час (из 9)
PF3	Основные структуры данных	6 осн. часов (из 14)
PF4	Рекурсия	2 осн. часа (из 5)
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL3	Основные вычислительные алгоритмы	3 осн. часа (из 12)
OS1	Обзор операционных систем	1 осн. час (из 2)
PL3	Введение в трансляцию	1 осн. час (из 2)
PL5	Механизмы абстракции	1 осн. час (из 3)
PL6	Объектно-ориентированное программирование	5 осн. часов (из 10)
IS1	Основы интеллектуальных систем	1 осн. час
IM1	Информационные модели и системы	1 осн. час (из 3)
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
SE3	Программные средства и окружения	1 осн. час (из 3)

Примечания:

Данный курс является третьим курсом в трехсеместровой последовательности ($CS101_B$-102_B-103_B), пытающейся дать общее интегрированное введение в информатику. Причины создания данной последовательности, а также предложения для данной реализации приведены в примечаниях к курсу $CS101_B$.

Б.1.5. "Алгоритмический" подход

Подход с ориентацией на алгоритмы (algorithms-first) существует только в двухсеместровом варианте. Если данный подход будет популярен, возможно, возникнет необходимость рассмотреть трехсеместровую реализацию.

$CS111_A$. Введение в алгоритмы и приложения

Данный курс начинает обзор вычислительных приложений и основ теории алгоритмов, состоящий из двух частей. В рамках этого курса студенты должны познакомиться с рядом понятий и конструкций теории алгоритмов, не зависящих от конкретного языка программирования, путем использования широкого круга прикладного программного обеспечения. Последующий курс $CS112_A$ показывает, как абстрактные понятия реализуются в контексте исполняемых программ.

Требования к слушателям: отсутствуют.

Описание курса:

- Основы: история технологии и мысли человека, включая технологию как катализатор изменения парадигм мышления; история программирования
- Алгоритмы и поиск решения задач: стратегии поиска решений задач; роль алгоритмов роль алгоритмов в процессе поиска решений задач; понятие и свойства алгоритма; описания алгоритмов при помощи псевдокода
- Знакомство с рекурсией: понятие рекурсии; рекурсивные математические функции; простые рекурсивные процедуры; стратегии "разделяй-и-властвуй"
- Базовые конструкции программирования: переменные, типы, выражения и присваивания; конструкции ветвления и итеративные конструкции управления; абстрагирование при помощи процедур и функций
- Основные структуры данных: типы; массивы; структуры; понятие абстрактного типа данных
- Введение в объектно-ориентированное программирование: объектно-ориентированное проектирование; инкапсуляция и скрытие информации; разделение интерфейса и реализации; классы, наследники и наследование; полиморфизм
- Основные вычислительные алгоритмы: простые численные алгоритмы; последовательный и дихотомический поиск; алгоритмы сортировки
- Основы анализа алгоритмов: введение в теорию вычислительной сложности; установление различий между лучшим, средним и худшим случаями; О-нотация; стандартные классы сложности; эмпирические измерения характеристик исполнения; затраты по объему используемой памяти и времени работы алгоритмов
- Основы теории вычислимости: легко разрешимые задачи и трудно разрешимые задачи; проблема останова; следствия невычислимости; границы применимости программирования

Затрагиваемые разделы:

PF1	Основные конструкции программирования	9 осн. часов
PF2	Алгоритмы и решение задач	3 осн. часа (из 6)
PF3	Основные структуры данных	6 осн. часов (из 14)
PF4	Рекурсия	3 осн. часа (из 5)
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL2	Алгоритмические стратегии	2 осн. часа (из 6)
AL3	Основные вычислительные алгоритмы	2 осн. часа (из 12)
AL5	Основы теории вычислимости	1 осн. час (из 6)
AL6	Классы сложности P и NP	1 осн. час
PL1	Обзор языков программирования	1 осн. час (из 2)
PL5	Механизмы абстракции	2 осн. часа (из 3)
PL6	Объектно-ориентированное программирование	4 осн. часа (из 10)
SP1	История информатики	1 осн. час
SE1	Проектирование ПО	2 осн. часа (из 8)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)

Примечания:

Программа данного курса состоит из трех частей:

- Курс знакомит студентов с понятиями и конструкциями теории алгоритмов безотносительно какого-либо конкретного языка программирования и без учета вопросов эффективного исполнения кода. Студенты учатся составлять и анализировать алгоритмы на уровне псевдокода, выполняя алгоритмы только вручную или в уме. Это позволяет студентам научиться различать существенные понятия и конструкции от специфических черт языка программирования. Отсутствие требований эффективного исполнения позволяет достаточно бы-

стро изучить понятия и конструкции характерные для функциональных, императивных и объектно-ориентированных парадигм.

- Параллельно с первой частью программы, вторая часть знакомит студентов с основными вычислительными приложениями, чтобы а) дать возможность студентам получить практический опыт программирования в дополнение к программированию на бумаге, предлагаемому в первой части; б) показать выразительную силу и необходимость абстракции в ситуациях, отличающихся от традиционного программистского контекста; в) заложить основы для использования студентами механизма абстракции в различных практических ситуациях.

- После первых двух частей, студенты должны приобрести достаточный опыт в осознанной разработке, трассировке и анализе алгоритмов. В третьей части, теоретические и практические аспекты предмета сливаются, позволяя студентам перейти к применению теоретических понятий и конструкций в современном производственном программном окружении.

Программа лекций и домашних заданий концентрируется на понятии абстракции, создании и анализе алгоритмов с использованием неисполняемого псевдокода. Программа практических занятий должна развить у студентов навыки использования программного обеспечения и собственно программирования, с особым упором на абстракцию как ключевую составляющую успешного использования языков программирования. Общая задача заключается в том, чтобы дать студентам широкую алгоритмическую базу, не зависящую от языков программирования. Таким образом, студенты получают соответствующую подготовку для последующего перехода к интенсивному курсу "Введение в программирование" с использованием любых языков и парадигм программирования.

$CS112_A$. Методология программирования

Данный курс основан на знаниях, заложенных в курсе $CS111_A$, и дает студентам возможность погрузиться в практику программирования и связанные с ней методы, с особым упором на объектно-ориентированнную парадигму. В этом курсе акцентируются вопросы, связанные с эффективной практикой программной инженерии, включая поэтапное проектирование, систематическое тестирование и отладку программных продуктов, основанную на методе доказательства гипотезы.

Требования к слушателям: $CS111_A$.

Описание курса:

- Обзор элементарного программирования и структур данных
- Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования; виртуальные машины
- Рекурсия: стратегии "разделяй-и-властвуй"; рекурсивный перебор с возвратами; деревья игр; реализация рекурсии
- Основные структуры данных: строки и операции со строками; представление данных в памяти; статическое, стековое размещения и размещение в куче; динамическое управление памятью; указатели и ссылки; связные структуры, стратегии реализации стеков, очередей, хэш-таблиц; стратегии реализации графов и деревьев; стратегии выбора правильной структуры данных
- Объектно-ориентированное программирование: обзор базовых понятий; объектно-ориентированное проектирование; иерархии классов; классы-коллекции и правила итерации; внутреннее представление объектов и таблицы методов
- Программирование событийно-управляемых систем: методы обработки событий; распространение событий; обработка исключений
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; стадии трансляции; машинно-зависимая и машинно-независимая части транслятора
- Алгоритмические стратегии: методы "грубой силы"; жадные алгоритмы; стратегии "разделяй-и-властвуй"; перебор с возвратами; метод ветвей и границ; эвристики; сравнение с образцом и алгоритмы, работающие со строками/текстом; алгоритмы численных приближений
- Основные вычислительные алгоритмы: хэш-таблицы; двоичные деревья поиска; представление графов; обходы в глубину и в ширину; алгоритмы кратчайшего пути; транзитивное замыкание; остовные деревья; топологическая сортировка; кучи
- Базовые методы программирования графики: использование графических API; графические пользовательские интерфейсы
- Введение в криптографию: исторический обзор криптографии; криптография с симметричным ключом и проблема обмена ключами; криптография с открытым ключом; цифровые подписи
- Методология разработки ПО: основные понятия и принципы проектирования; структурная декомпозиция; стратегии тестирования и отладки; разработка тестовых наборов; среды разработки; программные средства тестирования и отладочные программные средства

Затрагиваемые разделы:

PF3	Основные структуры данных	5 осн. часов (из 14)
PF4	Рекурсия	2 осн. часа (из 5)
PF5	Программирование событийно-управляемых систем	3 осн. часа (из 4)
AL2	Алгоритмические стратегии	2 осн. часа (из 6)
AL3	Основные вычислительные алгоритмы	4 осн. часа (из 12)
AL9	Алгоритмы криптографии	2 часа
PL1	Обзор языков программирования	1 осн. час (из 2)
PL2	Виртуальные машины	1 осн. час
PL3	Введение в трансляцию	2 осн. часа
PL4	Описание и типы данных	3 осн. часа
PL5	Механизмы абстракции	1 осн. час (из 3)
PL6	Объектно-ориентированное программирование	4 осн. часа (из 10)
GV1	Основы методов программирования графики	2 осн. часа
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
SE3	Программные средства и окружения	2 осн. часа (из 3)
SE6	Проверка соответствия ПО	1 осн. час (из 3)
SE7	Эволюция программ	1 осн. час (из 3)

Примечания:

Как отмечалось в описании курса $CS111_A$, в том курсе студентам представлен полный объем алгоритмических понятий и конструкций. Благодаря этому, данный курс может сконцентрироваться на развитии у студентов навыков эффективного программирования, а именно сформировать у них систематический подход к проектированию, реализации, тестированию и отладке. Также курс позволяет студентам более быстро усваивать материал, чем при отсутствии соответствующей широкой алгоритмической подготовки.

Студенты, переведенные на информатику с другой специальности и прослушавшие ранее только один курс по информатике, отличающийся от $CS111_A$, скорее всего, будут испытывать трудности при изучении этого курса из-за недостаточной подготовленности. В таких случаях студентам следует рекомендовать изучить материал курса $CS111_A$ перед изучением данного курса. С другой стороны, студенты, успешно освоившие курсы CS111 и CS112, скорее всего, преуспеют при изучении данного курса и оценят его направленность на развитие навыков эффективного программирования.

Б.1.6. "Аппаратный" подход

Подход с ориентацией на аппаратную часть (hardware-first) существует только в двухсеместровой форме. Если данный подход вызовет интерес, возможно, появится необходимость рассмотрения трехсеместровой реализации.

$CS111_H$. Знакомство с компьютером

Данный курс предлагает изучать компьютер снизу-вверх, начав с битов и доходя по иерархии понятий до высокоуровневых языков.

Требования к слушателям: отсутствуют.

Описание курса:

- Введение: коротко об истории программирования; компоненты компьютерной системы
- Представление данных в памяти компьютера: биты, байты, слова; представление числовых данных и основания систем счисления; знаковые представления и представления в дополнительном коде; базовые операции с битами; представление нечисловых данных
- Цифровая логика: переключающие схемы; вентили; память
- Организация машины на уровне ассемблера: организация фон-неймановской машины; блок управления; выборка, дешифрация и выполнение инструкции; системы команд и типы команд; программирование на языке ассемблера; форматы инструкций
- Алгоритмы и поиск решений: стратегии решения задач; роль алгоритмов в процессе решения задач; понятие и свойства алгоритма
- Ввод и вывод: простейший ввод-вывод; файлы
- Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования; роль трансляции в процессе программирования

- Основные конструкции программирования: синтаксис и семантика языка высокого уровня; переменные, типы, выражения и присваивания; простейший ввод/вывод; конструкции ветвления и итеративные конструкции; функции и передача параметров; структурная декомпозиция
- Основные структуры данных: примитивные типы; массивы; структуры; строки и операции со строками; представление данных в памяти; указатели и ссылки
- Рекурсия: понятие рекурсии; рекурсивные математические функции; простые рекурсивные процедуры; реализация рекурсии
- Методология разработки ПО: основные принципы и понятия проектирования ПО; структурное проектирование; стратегии тестирования и отладки; разработка тестовых наборов; среды разработки; программные средства тестирования и отладочные программные средства

Затрагиваемые разделы:

PF1	Основные конструкции программирования	5 осн. часов (из 9)
PF2	Алгоритмы и решение задач	2 осн. часа (из 6)
PF3	Основные структуры данных	5 осн. часов (из 14)
PF4	Рекурсия	5 осн. часов
AL2	Алгоритмические стратегии	2 осн. часа (из 6)
AL3	Основные вычислительные алгоритмы	2 осн. часа (из 12)
AL5	Основы теории вычислимости	1 осн. час (из 6)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL4	Описание и типы данных	1 осн. час (из 3)
PL5	Механизмы абстракции	2 осн. часа (из 3)
AR1	Цифровая логика и цифровые системы	3 осн. часа (из 6)
AR2	Представление данных в памяти компьютера	2 осн. часа (из 3)
AR3	Организация машины на уровне ассемблера	2 осн. часа (из 9)
AR4	Организация памяти	2 осн. часа (из 5)
SP1	История информатики	1 осн. час
SE1	Проектирование ПО	2 осн. часа (из 8)
SE3	Программные средства и окружения	1 осн. час (из 3)
SE6	Проверка соответствия ПО	1 осн. час (из 3)

$CS112_H$. Объектно-ориентированные методы программирования

Данный курс расширяет знания, приобретенные студентами в курсе $CS111_H$, понятиями из области объектно-ориентированного программирования и алгоритмического анализа.

Требования к слушателям: $CS111_H$.

Описание курса:

- Обзор понятий программирования
- Алгоритмы и поиск решения задач: стратегии реализации алгоритмов; стратегии отладки
- Объектно-ориентированное программирование: объектно-ориентированное проектирование; инкапсуляция и скрытие информации; разделение интерфейса и реализации; классы, наследники и наследование; полиморфизм; иерархии классов; классы-коллекции и правила итерации; основные шаблоны проектирования
- Основные структуры данных: статическое, стековое размещения и размещение в куче; динамическое управление памятью; связные структуры; стратегии реализации стеков, очередей, хэш-таблиц, графов и деревьев; стратегии выбора правильной структуры данных
- Программирование событийно-управляемых систем и параллелизма: методы обработки событий; распространение событий; управление параллелизмом при обработке событий; обработка исключений
- Использование программных интерфейсов приложений: программирование с использованием API; программы просмотра классов и другие подобные инструменты; программирование на примерах; отладка программы, использующей программный интерфейс приложения
- Основы анализа алгоритмов: асимптотический анализ наибольшей и средней сложности; установление различий между лучшим, средним и худшим случаями; O-большое и o-малое нотации; стандартные классы сложности; эмпирические измерения характеристик исполнения; затраты по объему памяти и времени; использование рекуррентных соотношений для анализа рекурсивных алгоритмов
- Алгоритмические стратегии: методы "грубой силы"; жадные алгоритмы; "разделяй-и-властвуй"; возвраты; эвристики
- Основные вычислительные алгоритмы: простые численные алгоритмы; последовательные и дихотомический поиск; алгоритмы сортировки
- Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования; роль трансляции в процессе программирования

- Основные вопросы проектирования языков программирования: основные принципы разработки языков программирования; цели разработки; виды эквивалентности типов; модели данных; модели конструкций управления; механизмы абстракции
- Виртуальные машины: понятие виртуальной машины; иерархия виртуальных машин; промежуточные языки; вопросы безопасности, возникающие при запуске кода на удаленной машине
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; стадии трансляции; машинно-зависимая и машинно-независимая части транслятора; задача трансляции как задача программной инженерии
- Основы теории вычислимости: легко разрешимые задачи и трудно разрешимые задачи; существование невычислимых функций
- Базовые методы программирования графики: иерархия графического ПО; использование графических программных интерфейсов приложения
- Проектирование ПО: основные понятия и принципы проектирования; архитектура программного продукта; структурное проектирование; объектно-ориентированный анализ и проектирование; проектирование компонент; проектирование для повторного использования
- Программные средства и окружения: программные окружения; программные средства тестирования

Затрагиваемые разделы:

PF1	Основные конструкции программирования	4 осн. часа (из 9)
PF2	Алгоритмы и решение задач	1 осн. час (из 6)
PF3	Основные структуры данных	6 осн. часов (из 14)
PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL2	Алгоритмические стратегии	2 осн. часа (из 6)
AL3	Основные вычислительные алгоритмы	4 осн. часа (из 12)
PL1	Обзор языков программирования	1 осн. час (из 2)
PL2	Виртуальные машины	1 осн. час
PL4	Описание и типы данных	2 осн. часа (из 3)
PL5	Механизмы абстракции	1 осн. час (из 3)
PL6	Объектно-ориентированное программирование	8 осн. часов (из 10)
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
SE3	Программные средства и окружения	1 осн. час (из 3)
SE5	Спецификации и требования к ПО	1 осн. час (из 4)

Б.2. Прочие курсы первого года обучения

Курсы в данном разделе размещены в порядке нумерации.

CS105. Дискретные структуры (1)

Данный курс знакомит студентов с основами дискретной математики и ее применениями в информатике. Цель данного курса – создать надежный теоретический фундамент для последующих курсов. Здесь обсуждаются функции, отношения, множества, простейшие методы доказательства, булева алгебра, логика высказываний, цифровая логика, элементарная теория чисел и основы комбинаторики.

Требования к слушателям: Математическая подготовка, позволяющая заниматься математикой на университетском уровне.

Описание курса:

- Знакомство с логикой и доказательствами; прямые доказательства; доказательства от противного; математическая индукция
- Основные объекты: функции (сюръекции, инъекции, инверсии, композиция отображений); отношения (рефлексивные, симметричные, транзитивные, эквивалентность); множества (диаграммы Венна, дополнения, декартовы произведения, степенные множества); принцип Дирихле; мощность и счетность
- Булева алгебра: логические значения; стандартные операции над логическими значениями; законы де Моргана
- Логика высказываний: логические связки; таблицы истинности; нормальные формы (конъюнктивная и дизъюнктивная); общезначимость
- Цифровая логика: логические вентили; триггеры; минимизация цепей
- Элементарная теория чисел: разложимость на множители; свойства простых чисел; наибольший общий делитель и наименьшее общее частное; алгоритм Эвклида; арифметические операции над остатками; китайская теорема об остатках

- Основы комбинаторики: комбинаторные объекты; принцип Дирихле; перестановки и подстановки; биномиальные коэффициенты

Затрагиваемые разделы:

DS1	Функции, отношения и множества	9 часов (6 осн. + 3)
DS2	Основы логики	5 осн. часов (из 10)
DS3	Методы доказательств	4 осн. часа (из 12)
DS4	Основы комбинаторики	9 часов (5 осн. + 4)
AR1	Цифровая логика и цифровые системы	3 осн. часа (из 6)
	Элементарная теория чисел	5 часов
	Темы по выбору	5 часов

Примечания:

Предлагаемая реализация курса по дискретным структурам (DS) разбивает материал на два курса: CS105 и CS106. Поскольку, в противоположность курсу CS115, материал разбит на две части, многим разделам посвящено больше времени, чем требуется в обязательном наборе. Кроме того, вариант из двух курсов включает дополнительные темы, снижая потребность в освещении этих тем в последующих курсах, таких как введение в анализ алгоритмов (CS210).

Хотя основная тема курса – это изложение собственно дискретной математики, курс имеет больше шансов на успех, если он будет содержать отступления, освещающие приложения излагаемого материала, особенно с использованием методов доказательств, логики и комбинаторики. Например, теория чисел может быть рассмотрена в рамках рассмотрения криптографии с открытым ключом, чтобы студенты, проявляющие интерес к информационным приложениям имели стимул к изучению теоретического материала.

CS106. Дискретные структуры (2).

Данный курс продолжает изложение дискретной математики, начатое в курсе CS105. Темы данного курса включают в себя логику предикатов, рекуррентные соотношения, графы, деревья, матрицы, вычислительную сложность, элементарную вычислимость и дискретную вероятность.

Требования к слушателям: CS105.

Описание курса:

- Обзор предыдущего курса
- Логика предикатов: применение квантора всеобщности и квантора существования; правила введения конъюнкции и дизъюнкции; ограничения логики предикатов
- Рекуррентные соотношения: основные формулы; элементарные методы решения
- Графы и деревья: основные определения; простейшие алгоритмы; стратегии обхода; методы доказательств; остовные деревья; приложения
- Матрицы: основные свойства; приложения
- Вычислительная сложность: порядковый анализ; стандартные классы сложности
- Элементарная вычислимость: счетность и несчетность; диагональный метод доказательства несчетности континуума; определения классов P и NP; простая демонстрация проблемы останова
- Дискретная вероятность: конечные вероятностные пространства; условная вероятность, вероятности независимых событий, формулы Байеса; случайные события; случайные целочисленные величины; математическое ожидание

Затрагиваемые разделы:

DS2	Основы логики	7 осн. часов (из 10)
DS3	Методы доказательств	8 осн. часов (из 12)
DS5	Графы и деревья	4 осн. часа
DS6	Дискретная вероятность	2 осн. часа (из 6)
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL5	Основы теории вычислимости	3 осн. часа (из 6)
AL6	Классы сложности P и NP	2 часа
	Матрицы	3 часа
	Темы по выбору	5 часов

Примечания:

Предлагаемая реализация курса дискретных структур (DS) разбивает материал на две части: CS105 и CS106. Для учебных программ, которые подразумевают ускоренное изложение данного материала, разработан курс CS115, в котором тот же материал изложен в одном курсе. Однако, последовательность из двух курсов включает некоторые дополнительные темы, в основном из области анализа алгоритмов и теории сложности (AL). В результате, в случае принятия учебным учреждением последовательности из двух курсов, во вводном курсе анализа алгоритмов (CS210) имеется можно уделить больше времени дополнительным вопросам.

Как и курс CS105, данный курс предполагает изложение математических тем в контексте их применения как инструментов информатики. Для данного курса характерными приложениями будут, например, задачи транспортных сетей (задача о коммивояжере) и задачи управления ресурсами.

CS115. Дискретные структуры в информатике

Данный курс знакомит студентов с основами дискретной математики, с упором на ее использование в информатике. Темы курса включают в себя функции, отношения и множества, логику высказываний и логику предикатов, схемную логику, методы доказательств, элементарную комбинаторику и дискретную вероятность.

Требования к слушателям: Математическая подготовка, позволяющая заниматься математикой на университетском уровне.

Описание курса:

- Основные объекты: функции (сюръекции, инъекции, инверсии, композиция отображений); отношения (рефлексивные, симметричные, транзитивные, эквивалентность); множества (диаграммы Венна, дополнения, декартовы произведения, степенные множества); принцип Дирихле; мощность и счетность
- Основы логики: логика высказываний; логические связки; таблицы истинности; нормальные формы (конъюнктивная и дизъюнктивная); достоверность формулы; логика предикатов; ограничения логики предикатов; применение квантора всеобщности и квантора существования; правила введения конъюнкции и дизъюнкции
- Цифровая логика: логические вентили; триггеры, счетчики; минимизация цепей
- Методы доказательств: понятия следствия, обратного утверждения, обращения утверждения, обратного следствия, отрицания и опровергающего утверждения; структура формальных доказательств; доказательство от противного; доказательство путем прихода к противоречию; математическая индукция; сильная индукция; рекурсивные математические определения; вполне упорядоченные множества
- Основы комбинаторики: комбинаторные объекты; принцип Дирихле; перестановки и подстановки; рекурсивные отношения
- Дискретная вероятность: конечные вероятностные пространства; условная вероятность, вероятности независимых событий, формулы Байеса; случайные события; случайные целочисленные величины; математическое ожидание

Затрагиваемые разделы:

DS1	Функции, отношения и множества	6 осн. часов
DS2	Основы логики	10 осн. часов
DS3	Методы доказательств	9 осн. часов (из 12)
DS4	Основы вычислений	5 осн. часов
DS6	Дискретная вероятность	6 осн. часов
AR1	Цифровая логика и цифровые системы	3 осн. часа (из 6)
	Темы по выбору	1 час

Примечания:

Данная реализация курса Дискретных Структур (DS) излагает основной материал за один курс. Хотя данная стратегия вполне состоятельна, многие учебные учреждения предпочтут использовать реализацию, включающую в себя два курса (представленную курсами CS105 и CS106), с целью более детального изложения материала.

CS120. Введение в устройство компьютера

Данный курс знакомит студентов с понятиями компьютеров и информационных систем, представляя процесс вычислений как работу иерархии виртуальных машин, начиная от аппаратуры и продвигаясь выше через все усложняющиеся уровни программного обеспечения. Данный курс описывает возможности, предоставляемые каждой виртуальной машиной, а также механизмы и инструменты, позволяющие реализовывать такую иерархию на практике.

Требования к слушателям: отсутствуют.

Описание курса:

- Основные элементы цифровой логики и их использование при создании компьютеров
- Описание работы компьютера на уровне регистров; функциональная организация компьютера
- Представление различных видов данных
- Элементы микропрограммирования и программирования на языке ассемблера
- Роль и функции языков программирования и их библиотек
- Роль и функции операционной системы (включая сети и распределенные системы)
- Приложения, включая описание их функциональных возможностей (текстовые процессоры, базы данных, браузеры, поисковые системы и т.д.)
- Взаимодействие человека и компьютера; значимость пользовательского интерфейса
- Знакомство с WWW: основы всемирной сети; браузеры; поисковые системы; информационный поиск; создание web-страниц
- Информация о сетях: информационные серверы; группы новостей; стратегии поиска; поиск и хранение информации; основополагающие принципы
- Вопросы интеллектуальной собственности

Затрагиваемые разделы:

AR2	Представление данных в памяти компьютера	1 осн. час (из 3)
AR3	Организация машины на уровне ассемблера	3 осн. часа (из 9)
AR6	Функциональная организация устройств	1 осн. час (из 7)
OS2	Основы операционных систем	1 осн. час (из 2)
NC1	Введение в распределенные вычисления	1 осн. час (из 2)
NC2	Телекоммуникации и сети	4 осн. часа (из 7)
NC4	WWW как пример клиент-серверного программирования	2 осн. часа (из 3)
PL3	Введение в трансляцию	1 осн. час (из 2)
HC1	Основы взаимодействия человека и машины	3 осн. часа (из 6)
HC2	Построение простого графического интерфейса	2 осн. часа
HC3	Оценка эргономичности ПО	3 часа
HC4	Создание эргономичного ПО	3 часа
IS1	Основы интеллектуальных систем	1 осн. час
IS2	Поиск решений	2 осн. часа (из 5)
IM1	Информационные модели и системы	2 осн. часа (из 3)
IM2	СУБД	1 осн. час (из 3)
SP1	История информатики	1 осн. час
SP2	Социальные вопросы программирования	2 осн. часа (из 3)
SP4	Профессиональная и этическая ответственность	1 осн. час (из 3)
SP6	Интеллектуальная собственность	1 осн. час (из 3)
	Темы по выбору	4 часа

Примечания:

Компьютерные системы невероятно сложны. Однако, взгляд на них как на иерархию абстрактных (виртуальных) машин позволяет раскрыть красоту и элегантность их конструкции. Такой подход дает возможность продемонстрировать наиболее важные аспекты информатики как научной дисциплины. Целью данного курса является рассмотрение различных распространенных виртуальных машин с целью сравнения предлагаемых ими возможностей и механизмов, а также изучение природы механизмов и программных средств, позволяющих реализовывать виртуальные машины различных уровней иерархии.

Изложение этого материала в интересной и захватывающей манере может показаться сложной задачей. Решением должно стать описание устройства компьютерной системы, соответствующее самым современным требованиям. Это поможет оправдать ожидания студентов и послужит стимулом для дальнейшего интереса к данной области. При этом необходимо подчеркнуть важность не только традиционных данных, но и звука, видеоизображения и т.д.

Благодаря огромному количеству ресурсов, доступных в WWW, данный модуль можно также использовать для обучения чрезвычайно широкому спектру навыков: научному поиску, умению представить результаты работы и т.д. В то же время, важно, чтобы студенты понимали проблемы, возникающие при работе с большими объемами информации.

Для решения своих собственных задач, каждому студенту придется работать с большим количеством сообщений электронной почты различной важности, документами и информацией различного сорта, ссылками на сайты, программными средствами, исследованиями, иллюстрациями, и т.д. При правильной организации и тщательном структурировании эти средства могут быть использованы для эффективной организации работы. Таким образом,

материал данного курса должен быть очень ценен для студентов. Обобщая, можно утверждать, что те же самые принципы могут быть использованы для создания окружений с самыми различными использованиями: для обучения, использования настольных издательских систем, управления проектами, поиска информации в глобальной сети, компьютерной графики и анимации, разработки компьютерных игр и т.д.

CS130. Введение в WWW

Данный курс знакомит студентов с миром информатики с помощью WWW, уделяя особое внимание способам создания web-страниц. Никакого предварительного опыта программирования не требуется, хотя к окончанию курса студенты приобретут некоторые навыки программирования на языках сценариев.

Требования к слушателям: отсутствуют.

Описание курса:

- Знакомство с Интернетом: история и истоки сетей и Интернета; обзор сетевых архитектур
- Телекоммуникации и сети: обзор сетевых стандартов и протоколов; сравнение коммутации цепи и пакетной коммутации
- Знакомство с WWW: web-технологии; протокол HTML; формат web-страницы; программные средства для создания web-сайтов
- Мультимедийные технологии: звук, изображения, анимация и видео; устройства ввода и вывода; программные средства для разработки мультимедийных приложений
- Интерактивность в WWW: языки сценариев; роль апплетов
- Взаимодействие человека и компьютера: аспекты человеко-машинного взаимодействия, влияющие на дизайн web-страниц; проектирование графического пользовательского интерфейса
- Управление сетями: обзор вопросов управления сетью; использование паролей и механизмов управления доступом; доменные имена и сервисы; вопросы ответственности поставщика сервисов Интернета; вопросы безопасности и брандмауэры
- Упаковка и распаковка данных: аналоговые и цифровые представления; обзор алгоритмов кодирования и декодирования; сжатие с потерями и сжатие без потерь
- Сетевая безопасность: основы криптографии; алгоритмы с секретным ключом; алгоритмы с открытым ключом; протоколы аутентификации; цифровые подписи; примеры
- Программные средства и окружения: программные средства разработки web-страниц
- Интеллектуальная собственность: основы интеллектуальной собственности; авторское право, патенты и профессиональная тайна; вопросы использования интеллектуальной собственности в сетях
- Право на частную жизнь и гражданские свободы: этические и правовые основания защиты конфиденциальности (privacy); свобода выражения в киберпространстве; международные и многонациональные особенности

Затрагиваемые разделы:

NC1	Введение в распределенные вычисления	2 осн. часа
NC2	Телекоммуникации и сети	2 осн. часа (из 7)
NC3	Сетевая безопасность	3 осн. часа
NC4	WWW как пример клиент-серверного программирования	3 осн. часа
NC5	Создание веб-приложений	3 часа
NC6	Управление сетями	2 часа
NC7	Сжатие и декодирование данных	3 часа
NC8	Мультимедийные технологии	3 часа
HC5	Проектирование графического пользовательского интерфейса	2 часа
HC7	Аспекты человеко-машинного взаимодействия в мультимедийных системах	2 часа
SE3	Программные средства и окружения	2 осн. часа (из 3)
SP6	Интеллектуальная собственность	2 осн. часа (из 3)
SP7	Конфиденциальность и гражданские свободы	2 осн. часа
	Темы по выбору	9 часов

Б.3. Основные курсы.

Несмотря на то, что курсы в данной части содержат указание на предпочтительную схему изложения (тематическую, сжатую, системную или web-ориентированную), все курсы имеют уникальные номера. Поэтому мы расположили курсы по порядку номеров. Если один и тот же курс присутствует более чем в одной схеме, для него указаны все возможные суффиксы.

CS210$_{\{C,S,T,W\}}$. Разработка и анализ алгоритмов

Данный курс знакомит студентов с формальными методами, используемыми для разработки и анализа алгоритмов. Курс концентрируется на лежащей в основе алгоритмов математической теории и вопросах практической эффективности. Темы курса включают в себя введение в теорию автоматов и их приложения в трансляции.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Обзор методов доказательств
- Основы анализа алгоритмов: асимптотический анализ максимальной и средней сложности; нахождение различий между лучшим, средним и худшим случаем; О-нотация; стандартные классы сложности; эмпирические измерения характеристик выполнения программы; компромиссы между алгоритмическим временем и алгоритмической памятью; использование рекуррентных соотношений для анализа рекурсивных алгоритмов
- Основные алгоритмические стратегии: методы "грубой силы"; жадные алгоритмы; "разделяй-и-властвуй"; возвраты; метод ветвей и границ; эвристики; сопоставление с образцом и алгоритмы, работающие со строками/тестом; алгоритмы численных приближений
- Основные структуры данных: стратегии реализации графов и деревьев; вопросы эффективности для структур данных
- Алгоритмы на графах и деревьях: обходы в ширину и в глубину; алгоритмы кратчайшего пути (алгоритмы Дейкстры и Флойда); транзитивное замыкание (алгоритм Флойда); минимальное остовное дерево (алгоритмы Прима и Краскала); топологическая сортировка
- Теория автоматов: конечные автоматы; машины Тьюринга; контекстно-свободные грамматики; невычислимые функции; проблема останова; следствия невычислимости
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; стадии трансляции; Машинно-зависимая и машинно-независимая части транслятора; промышленные трансляторы

Затрагиваемые разделы:

DS3	Методы доказательств	3 осн. часа (из 12)
DS5	Графы и деревья	4 осн. часа
PF2	Алгоритмы и решение задач	3 осн. часа
PF3	Основные структуры данных	3 осн. часа (из 14)
PL3	Введение в трансляцию	2 осн. часа
AL1	Базовый анализ алгоритмов	2 осн. часа (из 4)
AL2	Алгоритмические стратегии	6 осн. часов
AL3	Основные вычислительные алгоритмы	6 осн. часов (из 12)
AL5	Основы теории вычислимости	6 осн. часов
AL6	Классы сложности P и NP	2 часа
AL7	Теория автоматов	2 часа
	Темы по выбору	1 час

Примечания:

Тема анализа алгоритмов является центральной в информатике. Главной задачей данного курса является освещение широкого круга классических алгоритмов, используемых для решения практических задач. Каждый алгоритм обладает как достоинствами, так и недостатками. Более того, эффективность каждого алгоритма обычно зависит от размера и природы входных данных. Студентам необходимо получить ясное представление о способах анализа алгоритмов, чтобы уметь выбрать правильный алгоритм для конкретной задачи.

Студенты особенно хорошо воспринимают материал данного курса, если они видят связь между теорией и практикой. Для достижения этой цели преподаватель должен сопровождать теоретические темы практическими занятиями. Также необходимо, чтобы преподаватель привел яркие примеры больших различий по времени исполнения алгоритмов с различными сложностными характеристиками.

Одним из мощных средств улучшения понимания студентами алгоритмов и их сложностных характеристик может стать анимация алгоритмов. Средства анимации классических алгоритмов широко распространены в Интернете. Эти средства дают наглядные свидетельства сложности алгоритмов и, тем самым, улучшают понимание теоретических результатов.

Допустим и более формальный подход к данной теме, уделяющий максимум внимания формальной спецификации алгоритмов и доказательствам корректности, возможно, с поддержкой соответствующими средствами формальной спецификации и верификации программ. Однако неформальный подход, видимо, покажется привлекательным более широкому кругу студентов.

Студенты, успешно прошедшие данный курс, должны уметь:

- Объяснять математические понятия, используемые при описании сложности алгоритма.
- В каждом конкретном случае выбирать и применять на практике подходящие алгоритмы.
- В каждом конкретном случае выбирать и использовать одну из стратегий разработки алгоритмов.
- Объяснять особенности каждого из набора алгоритмов с одинаковой функциональностью.

$CS220_{\{C,S,T\}}$. Архитектура ЭВМ

Данный курс знакомит студентов с организацией и архитектурой компьютерных систем, начиная со стандартной фон неймановской модели и заканчивая новейшими понятиями в архитектуре ЭВМ.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Цифровая логика: основные составные блоки (логические вентили, триггеры, счетчики, регистры, программируемые логические матрицы); логические выражения, минимизация, сумма мультипликативных форм; нотация пересылки регистров; физические аспекты (задержки вентилей, нагрузочные модули по входу и по выходу)
- Представление данных: биты, байты, слова; представление числовых данных и основания систем счисления; системы с фиксированной и с плавающей точкой; знаковые представления и представления в дополнительном коде; представление нечисловых данных (коды символов, графические данные); представление структур и массивов
- Организация машины на уровне ассемблера: основы организации фон-неймановской машины; управляющее устройство; выборка, дешифрация и выполнение команд; системы команд и типы команд (обработка данных, управляющие, ввод-вывод); программирование на языке ассемблера; форматы инструкций; режимы адресации; механизмы вывоза подпрограммы и возврата из них; ввод-вывод и прерывания
- Организация памяти: системы хранения и их технология; кодирование, сжатие данных и целостность данных; иерархия памяти; организация и функции основной памяти; латентность, время цикла, полоса пропускания и чередование; кэш-память (преобразование адресов, размер блока, политика замещения и сохранения); виртуальная память (таблица страниц, TLB); обработка ошибок доступа к памяти и надежность
- Организация взаимодействия устройств: основы ввода-вывода; установление связи, буферизация, программируемый ввод-вывод, ввод-вывод по прерыванию; структура прерываний; направленная и приоритетная обработка прерываний; внешние хранители, физическая организация и диски; шины: протоколы, организация доступа к общей шине, прямой доступ к памяти; знакомство с сетями; поддержка мультимедиа; RAID-архитектуры
- Функциональная организация устройств: реализация схем с простой передачей данных; управляющее устройство; сравнение аппаратной и микропрограммной реализаций; водопровод; знакомство с распараллеливанием инструкций
- Многопроцессорные и альтернативные архитектуры: знакомство с SIMD, MIMD, VLIW, EPIC; систолическая архитектура; сети с внутрисхемной коммутацией; архитектуры с разделяемой памятью; координирование работы кэша; модели и целостность памяти
- Увеличение производительности устройств: RISC-архитектура; предсказывание переходов; выборка команд с упреждением; модульное наращивание
- Современные архитектуры: "карманные" компьютеры; встроенные устройства; направления развития архитектуры процессоров

Затрагиваемые разделы:

AR1	Цифровая логика и цифровые системы	3 осн. часа (из 6)
AR2	Представление данных в памяти компьютера	3 осн. часа
AR3	Организация машины на уровне ассемблера	9 осн. часов
AR4	Организация памяти	5 осн. часов
AR5	Организация взаимодействия устройств	3 осн. часа

AR6	Функциональная организация устройств	7 осн. часов
AR7	Многопроцессорные и альтернативные архитектуры	3 осн. часа
AR8	Увеличение производительности устройств	3 часа
	Современные архитектуры	2 часа
	Темы по выбору	2 часа

Примечания:

Различия во внутренней структуре и организации компьютеров приводят к значительным различиям в их производительности и функциональных возможностях. Это порождает невероятное разнообразие вычислительных устройств, начиная с "карманных" компьютеров и заканчивая высокопроизводительными машинами. Данный курс знакомит студентов с различными аспектами устройства компьютерной системы, соображениями проектирования компьютерной системы и компромиссами процесса создания компьютерной системы.

Ключевым моментом в данном курсе является мотивация. Необходимо увеличить интерес студентов и преподавателей к проектированию компьютеров, представить эту область исследований как интересную, важную и многообещающую. Один из подходов состоит во включении в данный курс существенной практической компоненты, в рамках которой студенты могли бы построить свою компьютерную систему. В этом случае студенты смогут ознакомиться с проектированием компьютеров значительно более детально, а также смогут ощутить чувство достижения в области архитектуры ЭВМ, похожее на то, которое они ощущают после завершения важного программного проекта.

Программные средства могут сыграть важную роль в данном курсе, особенно в тех случаях, когда бюджет университета не позволяет проводить лабораторные работы с аппаратурой. Перечислим наиболее полезные программные средства: имитаторы набора команд, эмуляторы кэша, системы оценки производительности (benchmarks) и т.д.

Студенты, успешно завершившие данный курс, должны уметь:

- Писать и отлаживать простые программы на языке ассемблера.
- Объяснять основополагающие принципы создания и развития компьютерных систем для различных назначений.
- Отслеживать влияние важных достижений в области программирования (таких как компиляторы, телекоммуникации, всемирная сеть, мультимедиа, безопасность) на архитектуру компьютерных систем.
- Перечислять архитектурные элементы современной компьютерной системы.

$CS221_W$. Архитектура ЭВМ и операционные системы

Данный курс знакомит студентов одновременно с понятиями архитектуры ЭВМ и операционных систем.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Цифровая логика и цифровые системы: основные составные блоки; логические выражения, минимизация, сумма мультипликативных форм; нотация пересылки регистров; физические аспекты
- Представление данных в памяти компьютера: биты, байты, слова; представление числовых данных и основания систем счисления; системы с фиксированной и с плавающей точкой; знаковые представления и представления в дополнительном коде; представление нечисловых данных; представление структур и массивов
- Организация машины на уровне ассемблера: основы архитектуры фон Неймана; управляющее устройство; выборка, дешифрация и выполнение команд; системы команд и типы команд; программирование на языке ассемблера; форматы инструкций; режимы адресации; механизмы вывоза подпрограмм и возврата из них; ввод-вывод и прерывания
- Организация памяти: системы хранения и их технология; кодирование, сжатие данных и целостность данных; иерархия памяти; организация и функции основной памяти; латентность, время цикла, полоса пропускания и чередование; кэш-память; виртуальная память; обработка ошибок доступа к памяти и надежность
- Функциональная организация устройств: реализация схем с простой передачей данных; управляющее устройство; водопровод; знакомство с распараллеливанием инструкций
- Обзор операционных систем: методы структуризации; абстракции, процессы и ресурсы; понятия интерфейсов прикладных программ; запросы приложения и эволюция программных и аппаратных методов; организация устройств; прерывания; понятие режима пользователя/супервизора и защита; переход в режим работы ядра

- Параллелизм: диспетчеризация и переключение контекстов; роль прерываний; параллельное исполнение; проблема взаимного исключения и некоторые решения
- Теория очередей: очереди с приоритетами и без приоритетов; планировщики и политики их работы; процессы и нити
- Управление памятью: обзор физической памяти и управляющей аппаратуры; оверлеи, подкачка и разделы; страничная организация памяти и сегментация; стратегии подкачки и выгрузки страниц; рабочие множества и пробуксовка; кэширование

Затрагиваемые разделы:

AR1	Цифровая логика и цифровые системы	3 осн. часа (из 6)
AR2	Представление данных в памяти компьютера	3 осн. часа
AR3	Организация машины на уровне ассемблера	9 осн. часов
AR4	Организация памяти	5 осн. часов
AR6	Функциональная организация устройств	7 осн. часов
OS1	Обзор операционных систем	2 осн. часа
OS2	Основы операционных систем	2 осн. часа
OS3	Параллелизм	2 осн. часа (из 6)
OS4	Планирование и диспетчеризация	2 часа (из 3)
OS5	Управление памятью	5 осн. часов

$CS222_W$. Архитектура сетей и коммуникаций.

Данный курс знакомит студентов с аспектами архитектуры компьютера, критичными для телекоммуникаций и сетей.

Требования к слушателям: $CS221_W$.

Описание курса:

- Распределенные алгоритмы: консенсус и выборы; обнаружение завершения; отказоустойчивость; стабилизация
- Организация взаимодействия устройств: основы ввода-вывода; структура прерываний; внешние хранители, физическая организация и диски; шины; знакомство с сетями; поддержка мультимедиа; RAID-архитектуры
- Многопроцессорные и альтернативные архитектуры: знакомство с SIMD, MIMD, VLIW, EPIC; систолическая архитектура; схемы с внутренней коммутацией; системы с разделяемой памятью; координирование работы кэша; модели памяти и целостность памяти
- Архитектура сетей и распределенных систем: знакомство с локальными и глобальными сетями; многоуровневое устройство протоколов, ISO/OSI, IEEE 802; влияние архитектурных решений на распределенные алгоритмы; сетевые вычисления; распределенные мультимедиа
- Параллелизм: состояния и диаграммы состояний; структуры; диспетчеризация и переключение контекстов; роль прерываний; параллельное исполнение; проблема взаимного исключения и некоторые решения; блокировки; модели и механизмы; проблемы поставщика-потребителя и синхронизация; особенности мультипроцессорных систем
- Планирование и диспетчеризация: обзор процессов и планирования; учет предельных сроков и реального времени
- Системы реального времени и встроенные системы: планирование процессов и задач; требования к управлению памятью и внешним носителем в системах реального времени; отказы, риски и восстановление; специальные вопросы систем реального времени
- Отказоустойчивость: основные понятия; временная и пространственная избыточность; методы, используемые для реализации отказоустойчивости; примеры надежных систем
- Оценивание производительности систем: причины необходимости оценивания производительности систем; политики кэширования, страничной организации, планирования, управления памятью, безопасности и т.д.; модели оценивания; способы сбора оценочной информации
- Программирование сценариев: программирование сценариев и роль языков сценариев; основные системные команды; создание сценариев; передача параметров; выполнение сценариев; влияние программирования сценариев на программирование в целом

Затрагиваемые разделы:

AL4	Распределенные алгоритмы	3 осн. часа
AR5	Организация взаимодействия устройств	3 осн. часа
AR7	Многопроцессорные и альтернативные архитектуры	3 осн. часа
AR9	Архитектура сетей и распределенных систем	5 часов

OS3	Параллелизм	4 осн. часа (из 6)
OS4	Планирование и диспетчеризация	2 осн. часа (из 3)
OS9	Системы реального времени и встроенные системы	5 часов
OS10	Отказоустойчивость	5 часов
OS11	Оценивание производительности систем	4 часа
OS12	Программирование сценариев	3 часа
	Темы по выбору	3 часа

CS225$_{\{S,T\}}$. Операционные системы

Данный курс знакомит студентов с основами устройства и реализации операционных систем. Темы курса включают обзор компонентов операционной системы, вопросы взаимного исключения и синхронизации, реализацию процессов, алгоритмы планирования, управление памятью и файловые системы.

Требования к слушателям: CS220.

Описание курса:

- Обзор операционных систем: роль и задачи операционных систем; история развития операционных систем; функциональность типичной операционной системы; вопросы, влияющие на устройство ОС (эффективность, устойчивость, гибкость, переносимость, безопасность, совместимость)
- Основные принципы работы операционных систем: методы структуризации; абстракции, процессы и ресурсы; создание программных интерфейсов приложений; организация устройств; прерывания; переключения между режимами работы пользователя/супервизора
- Параллелизм: понятие параллельного исполнения; состояния и диаграммы состояний; структуры, используемые реализацией параллелизма (таблицы готовности, блоки управления процессом и т.д.); диспетчеризация и переключение контекстов; обработка прерываний при наличии параллельного исполнения
- Взаимное исключение: описание проблемы взаимного исключения; обнаружение и предотвращение блокировок; стратегии решения проблемы; модели и механизмы (семафоры, мониторы, переменные состояния, рандеву); задача поставщика-потребителя; синхронизация; особенности мультипроцессорных систем
- Планирование: вытесняющее и невытесняющее планирование; политики планирования; процессы и нити; особенности систем реального времени
- Управление памятью: обзор физической памяти и управляющей аппаратуры; оверлеи, подкачка и разделы; страничная организация памяти и сегментация; стратегии подкачки и выгрузки страниц; рабочие множества и пробуксовка; кэширование
- Управление устройствами: характеристики последовательных и параллельных устройств; абстрактные понятия различий устройств; стратегии буферизации; прямой доступ к памяти; восстановление после сбоев
- Файловые системы: основные понятия (данные, метаданные, операции, организация, буферизация, последовательные файлы и файлы с непоследовательным размещением); содержание и структура каталогов; методы работы файловой системы (сегментирование дисковой памяти, монтирование и демонтирование, виртуальные файловые системы); файлы, отображаемые в память; файловые системы специального назначения; именование, поиск и доступ; стратегии резервного копирования
- Безопасность и защита в ОС: обзор системы безопасности ОС; разделение политики безопасности и механизма ее реализации; методы и устройства обеспечения безопасности; защита, доступ и аутентификация; модели защиты; защита памяти; шифрование; управление восстановлением

Затрагиваемые разделы:

AL4	Распределенные алгоритмы	3 осн. часа
OS1	Обзор операционных систем	2 осн. часа
OS2	Основы операционных систем	2 осн. часа
OS3	Параллелизм	6 осн. часов
OS4	Планирование и диспетчеризация	3 осн. часа
OS5	Управление памятью	5 осн. часов
OS6	Управление устройствами	4 часа
OS7	Безопасность и защита в ОС	4 часа
OS8	Файловые системы	5 часов
OS11	Оценивание производительности систем	2 часа
	Темы по выбору	4 часа

CS226$_{\{C,S\}}$. Операционные системы и сети

Данный курс знакомит студентов с основами операционных систем, а также с основами сетей и телекоммуникаций.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Знакомство с программированием событийно-управляемых систем
- Использование программных интерфейсов приложения (API): программирование с использованием API; программы просмотра классов и другие подобные инструменты; программирование с помощью примеров; отладка программы, использующей API
- Обзор операционных систем: роль и задачи операционных систем; история развития операционных систем; функциональность типичной операционной системы
- Основные принципы работы операционных систем: методы структуризации; абстракции, процессы и ресурсы; понятие программных интерфейсов приложений; организация устройств; прерывания; понятия режимов работы пользователя/супервизора и защиты
- Введение в параллелизм: принципы синхронизации; проблема взаимного исключения и некоторые решения; избегание блокировок
- Параллелизм: состояния и диаграммы состояний; структуры; диспетчеризация и переключение контекстов; роль прерываний; параллельное исполнение; проблема взаимного исключения и некоторые решения; блокировки; модели и механизмы; проблемы поставщика-потребителя и синхронизация
- Планирование и диспетчеризация: вытесняющее и невытесняющее планирование; планировщики и политики их работы; процессы и нити; учет предельных сроков и реального времени
- Управление памятью: обзор физической памяти и управляющей аппаратуры; оверлеи, подкачка и разделы; страничная организация памяти и сегментация; стратегии подкачки и выгрузки страниц; рабочие множества и пробуксовка; кэширование
- Введение в распределенные алгоритмы: консенсус и выборы; отказоустойчивость
- Введение в распределенное программирование: причины возникновения и история сетевого программирования и Интернета; сетевые архитектуры; круг специализаций в сетевом программировании
- Введение в телекоммуникации и сети: сетевые архитектуры; вопросы, связанные с распределенными вычислениями; простые сетевые протоколы; API для работы выполнения сетевых операций
- Знакомство с WWW: web-технологии; характеристики web-серверов; природа связи клиент-сервер; web-протоколы; программные средства для создания и управления web-сайтом
- Сетевая безопасность: основы криптографии; алгоритмы с секретным ключом; алгоритмы с открытым ключом; протоколы аутентификации; цифровые подписи; примеры

Затрагиваемые разделы:

PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
AL4	Распределенные алгоритмы	3 осн. часа
OS1	Обзор операционных систем	2 осн. часа
OS2	Основы операционных систем	2 осн. часа
OS3	Параллелизм	6 осн. часов
OS4	Планирование и диспетчеризация	3 осн. часа
OS5	Управление памятью	5 осн. часов
NC1	Введение в распределенные вычисления	2 осн. часа
NC2	Телекоммуникации и сети	7 осн. часов
NC3	Сетевая безопасность	3 осн. часа
NC4	WWW как пример клиент-серверного программирования	3 осн. часа
PL6	Объектно-ориентированное программирование	2 осн. часа (из 10)

Примечания:

Так как данный курс включает темы из различных областей, CS226 – это типичный пример "поперечного" (cross-cutting) подхода при составлении основного курса. Более традиционный подход подразумевает преподавание самостоятельных курсов по операционным системам и сетям. Однако данные темы сильно пересекаются, и потому имеет смысл создать курс, который обращался бы к этим двум темам одновременно, особенно учитывая большую популярность Интернета среди студентов. Совмещенное обсуждение операционных систем и сетей увеличивает мотивацию студентов и стимулирует их к осмыслению влияния всемирной сети на операционные системы, а также прочих более общих принципов.

Вопрос мотивации является главным при создании этого курса. Область операционных систем достаточно часто считается сложной – как для студентов, так и для преподавательского состава. Тем не менее, в данной области есть много идей, важных для всех специалистов. Поэтому преподаватели должны сделать изучение операционных систем как можно более интересным – именно это соображение должно быть решающим в выборе подхода к

преподаванию и обучению. Студенты должны увидеть взаимосвязь операционных систем с системами, которые они используют. Например, студентам можно предложить рассмотреть влияние на операционную систему таких областей исследований как телекоммуникации, мультимедиа, безопасность и карманных компьютеров. Аналогично, можно попросить студентов описать влияние на операционную систему следующих вопросов:

- Прослушивание звукового компакт-диска при использовании компьютера
- Загрузки телевизионных программ в окна
- Стыковка с компьютером таких устройств как цифровые камеры или карманные компьютеры
- Клиент-серверная архитектура

Целью любого курса по операционным системам является ознакомление студентов со многими важными идеями, находящими применение в операционных системах и сетях. Поэтому стоит особо остановиться на следующих соображениях:

- Идея кэширования, важная на уровне аппаратуры, применяется и в сетевых приложениях в контексте кэширования информации, загружаемой из всемирной сети.
- Понятия, возникающие при обсуждении виртуальной памяти, возникают снова при создании виртуальных окружений.
- Материал, относящийся к параллелизму, применим в обобщенном контексте программирования распределенных вычислений.
- Материал о распределении ресурсов и планировании оказывается главной составляющей частью исследования операций.
- Большая часть материалов курса имеет непосредственное отношение к разработке систем реального времени и надежных систем.

Студенты, вероятно, проявят больший интерес к операционным системам, если им доведется поработать в контексте реальной системы, а не какого-то абстрактного упрощения. В этом смысле ПО с открытым кодом (open-source movement) сделало неоценимый вклад в педагогический процесс, так как теперь исходные тексты многих известных операционных систем доступны бесплатно. Эти общедоступные ресурсы помогают иллюстрировать особенности операционных систем, а также дают полезные примеры разных реализаций некоторых функций. Нужно отметить, что многие студенты будут вдохновлены самой идеей установки, например, ОС Linux на их машины.

Студенты, успешно прошедшие данный курс, должны уметь:

- Излагать принципы, лежащие в основе устройства типичной операционной системы, демонстрируя также понимание более широкой применимости идей и влияния таких вещей как высокоуровневые языки, телекоммуникации, мультимедиа и вопросы безопасности.
- Использовать возможности операционной системы для решения простых задач, включая расширение функциональности путем интеграции новых компонентов.
- Знать проблемы безопасности, связанные с распределенными приложениями, работающими во всемирной сети, и уметь выбирать механизмы, решающие эти проблемы.

$CS230_{\{T,W\}}$. Распределенные вычисления

Данный курс знакомит студентов со структурой, реализацией и теоретическими основаниями распределенных вычислений, а также с приложениями этой технологии.

Требования к слушателям: $CS222_W$ или $CS225_T$.

Описание курса:

- Телекоммуникации и сети: сетевые стандарты и органы стандартизации; семиуровневая модель ISO и ее реализация в TCP/IP; сравнение коммутации каналов и пакетной коммутации; потоки и дейтаграммы; концепции физического уровня; понятия канального уровня; объединение сетей и маршрутизация; сервисы транспортного уровня
- WWW как пример клиент-серверного программирования: web-технологии; характеристики web-серверов; роль клиентских машин; природа связи клиент-сервер; web-протоколы; программные средства для создания и управления web-сайтом; разработка информационных серверов Интернета; размещение информации и приложений
- Создание web-приложений: протоколы прикладного уровня; принципы конструирования всемирной сети; сервера баз данных; удаленные вызовы процедур; легковесные распределенные объекты; роль промежуточного ПО; средства поддержки; вопросы безопасности в распределенных системах; корпоративные сетевые приложения

- Управление сетями: обзор вопросов управления сетями; вопросы сферы ответственности поставщика сервисов Интернета; вопросы безопасности и брандмауэры; вопросы качества сервиса
- Сжатие и декодирование данных: обзор основ сжатия данных; сжатие и декодирование звука; сжатие и декодирование изображений; сжатие и декодирование видео; соображения производительности
- Технологии работы с мультимедиа: обзор технологий работы с мультимедиа; стандарты мультимедиа; вопросы планирования загрузки и производительности; устройства ввода-вывода; MIDI-клавиатуры; синтезаторы; стандарты форматов хранения; серверы мультимедиа и файловые системы; программные средства, используемые при разработке мультимедиа
- Программирование мобильных систем: обзор истории, эволюции и совместимости стандартов мобильных устройств; специальные вопросы программирования мобильных устройств; беспроводные локальные сети и спутниковые сети; замкнутые беспроводные контуры; протоколы мобильного Интернета; адаптация для мобильных устройств; расширение архитектуры клиент-сервер для случая мобильных устройств; доступ к данным с мобильного устройства; средства разработки ПО для мобильных устройств; роль промежуточного ПО и программных средств поддержки; вопросы производительности; изменяющиеся технологии

Затрагиваемые разделы:

PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
NC1	Введение в распределенные вычисления	2 осн. часа
NC2	Телекоммуникации и сети	7 осн. часов
NC3	Сетевая безопасность	3 осн. часа
NC4	WWW как пример клиент-серверного программирования	3 осн. часа
NC5	Создание веб-приложений	8 часа
NC6	Управление сетями	2 часа
NC7	Сжатие и декодирование данных	3 часа
NC8	Мультимедийные технологии	3 часа
NC9	Программирование мобильных устройств	4 часа
PL6	Объектно-ориентированное программирование	2 осн. часа (из 10)
	Темы по выбору	1 час

CS240$_S$. Трансляция языков программирования

Данный курс знакомит студентов с теорией и практикой трансляции языков программирования. Темы курса включают в себя устройство компиляторов, лексический анализ, синтаксический анализ, таблицы символов, обработку объявлений, управление памятью, генерацию кода и методы оптимизации.

Требования к слушателям: CS210, CS220.

Описание курса:

- Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования; роль трансляции в процессе программирования
- Основные вопросы проектирования языков программирования: основные принципы разработки языков программирования; цели разработки; виды эквивалентности типов; модели данных; модели конструкций управления; механизмы абстракции
- Виртуальные машины: понятие виртуальной машины; иерархия виртуальных машин; промежуточные языки
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; стадии трансляции; машинно-зависимая и машинно-независимая части транслятора; трансляция как одна из задач программной инженерии
- Лексический анализ: применение регулярных выражений в программах лексического анализа; ручное кодирование и автоматическая генерация лексических анализаторов; формальное определение лексем; реализация конечного автомата
- Синтаксический анализ: формальное определение грамматик; BNF и EBNF; нисходящий и восходящий анализ; табличные синтаксические анализаторы и метод рекурсивного спуска; управление таблицами символов; использование средств поддержки процесса трансляции
- Модели управления выполнением: порядок вычисления подвыражений; исключения и их обработка; системы динамической поддержки
- Работа с объявлениями, модульностью и управление размещением в памяти: виды объявлений; механизмы параметризации; параметризация типов; механизмы разделения и ограничения областей видимости; сборка мусора
- Системы типов: тип данных как набор значений с операциями над ними; типы данных; модели проверки типов; семантические модели типов, определяемых пользователем; параметрический полиморфизм; полиморфизм подтипа; алгоритмы проверки типов

- Интерпретация: итеративная и рекурсивная интерпретации; итеративная интерпретация промежуточного представления; рекурсивная интерпретация дерева разбора программы
- Генерация кода: промежуточное представление и объектный код; промежуточные представления; реализация генераторов кода; генерация кода путем обхода дерева; контекстно-зависимая трансляция; использование регистров
- Оптимизация: машинно-независимая оптимизация; анализ потоков данных; оптимизации циклов; машинно-зависимая оптимизация

Затрагиваемые разделы:

PL1	Обзор языков программирования	2 осн. часа
PL2	Виртуальные машины	1 осн. час
PL3	Введение в трансляцию	2 осн. часа
PL8	Системы автоматического перевода языков	15 часов
PL9	Системы типов	4 часа
	Темы по выбору	16 часов

Примечания:

Данный курс имеет две различные, но взаимосвязанные цели. Во-первых, курс содержит вопросы теории трансляции. Во-вторых, материал курса показывает, как применять эту теорию при создании компиляторов и интерпретаторов, а также генераторов компиляторов. Курс освещает как создание компиляторов с нуля, так и использование генераторов компиляторов. Кроме того, курс обсуждает основные вопросы устройства компиляторов.

Как и со многими другими курсами по информатике, имеющими значительную теоретическую составляющую, инструменты визуализации могут значительно улучшить качество лекций и послужить анимированными комментариями к лекциям.

Создание компилятора/интерпретатора является необходимой составляющей курса, с помощью которой студенты смогут получить необходимые навыки. Однако, проекты по созданию компиляторов зачастую трудны из-за следующих проблем:

- Объем компилятора обычно превышает размеры каких-либо проектов, создававшихся студентами ранее.
- Большинство генераторов компиляторов используют табличный подход, что создает трудности при отладке конечного компилятора.

Сложность этих проблем может быть уменьшена путем использования декларативных лексических анализаторов, а также генераторов синтаксических анализаторов, использующих метод рекурсивного спуска.

$CS250_W$. Взаимодействие человека и машины

Данный курс знакомит студентов с принципами и методами взаимодействия человека и компьютера.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Основы человеко-машинного взаимодействия: мотивация; контексты взаимодействия человека и компьютера; создание и оценка эргономичных систем; модели поведения человека; вопросы учета человеческого разнообразия; принципы хорошего дизайна и хороших дизайнеров; технические ограничения; основы тестирования эргономичности ПО
- Оценка эргономичности ПО: выделение целей тестирования; оценка без участия пользователей; оценка с участием пользователей
- Разработка эргономичных систем: подходы, характеристики и обзор процесса; функциональные возможности и удобство использования; спецификация взаимодействия с пользователем и представления материала; методы и инструменты прототипирования
- Разработка графического интерфейса: выбор стилей и методов взаимодействия; отражение аспектов человеко-машинного взаимодействия в стандартных графических элементах управления; аспекты человеко-машинного взаимодействия в разработке внешнего вида экрана; обработка человеческих ошибок; более сложные методы дизайна экрана; мультимодальное взаимодействие; трехмерное взаимодействие и виртуальная реальность
- Программирование графического интерфейса: независимость диалогов и уровни анализа; классы графических элементов управления; обработка событий и взаимодействие с пользователем; управление конфигура-

цией; построители графических интерфейсов пользователя; окружения для программирования пользовательского интерфейса; межплатформенная разработка

- Аспекты человеко-машинного взаимодействия в системах мультимедиа: категоризация и архитектура информации; информационный поиск и поведение человека; эргономичная структура систем мультимедийной информации; распознавание речи и обработка естественного языка; информационные устройства и программирование мобильных устройств
- Аспекты человеко-машинного взаимодействия в системах коллективного пользования: программные средства автоматизации коллективной работы для специальных целей; асинхронное групповое взаимодействие; синхронное групповое взаимодействие; сообщества пользователей online-систем; ПО как деятельный субъект и интеллектуальные агенты

Затрагиваемые разделы:

PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
HC1	Основы взаимодействия человека и машины	6 осн. часов
HC2	Построение простого графического интерфейса	2 осн. часа
HC3	Оценка эргономичности ПО	5 часов
HC4	Создание эргономичного ПО	5 часов
HC5	Проектирование графического пользовательского интерфейса	6 часов
HC6	Программирование графического пользовательского интерфейса	3 часа
HC7	Аспекты человеко-машинного взаимодействия в мультимедийных системах	5 часов
HC8	Аспекты человеко-машинного взаимодействия в системах коллективного пользования	3 часа
PL6	Объектно-ориентированное программирование	2 осн. часа (из 10)
	Темы по выбору	1 час

$CS250_{\{S,W\}}$. Компьютерная графика

Данный курс знакомит студентов с основами компьютерной графики, которая становится все более важной областью в информатике. Компьютерная графика, особенно применительно к мультимедийным аспектам WWW, открыла новые возможности в области интерфейсов взаимодействия человека и компьютера. Целью данного курса является рассмотрение принципов, методов и программных средств, сделавшими возможными эти продвижения.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Графические системы: растровые и векторные графические системы; устройства отображения видеоинформации; физические и логические устройства ввода; проблемы, с которыми сталкивается разработчик графической системы
- Основные методы в графике: иерархия графических программных средств; использование графических API; простые модели цвета; однородные координаты; аффинные преобразования; трансформация отсечение
- Графические алгоритмы: алгоритмы генерации линий; структура и использование шрифтов; параметрические полиномиальные кривые и поверхности; полигональное представление трехмерных объектов; введение в трассировку лучей; синтезирование изображений, методы семплирования и сглаживания; улучшение изображений
- Принципы человеко-машинного взаимодействия: мотивация; создание и оценка эргономичных систем
- Проектирования графического интерфейса пользователя: выбор стилей и техники взаимодействия с пользователем; человеко-машинные аспекты проектирования; динамика цвета; структурирование информации для улучшения понимания
- Программирование графического интерфейса пользователя: экранные элементы; обработка событий и взаимодействие с пользователем; построители графических интерфейсов пользователя
- Компьютерная мультипликация: покадровая анамиция; анимация камеры; система сценариев; анимация сочлененных структур; захват движения; процедурная анимация; деформация
- Технологии работы с мультимедиа: аудио, видео и графика; устройство систем мультимедиа; программные средства для разработки приложений мультимедиа; виртуальная реальность

Затрагиваемые разделы:

AL10	Алгоритмы геометрических построений	2 часа
HC2	Построение простого графического интерфейса	2 осн. часа

HC3	Оценка эргономичности ПО	2 часа
HC4	Создание эргономичного ПО	2 часа
HC5	Проектирование графического пользовательского интерфейса	5 часов
HC6	Программирование графического пользовательского интерфейса	5 часов
GV1	Основы методов программирования графики	2 осн. часа
GV2	Графические системы	1 осн. час
GV3	Передача графических данных	2 часа
GV4	Геометрическое моделирование	3 часа
GV5	Основы визуализации	3 часа
GV8	Компьютерная мультипликация	2 часа
GV10	Виртуальная реальность	2 часа
IM13	Мультимедийная информация и системы	4 часа
SE2	Использование программных интерфейсов приложения	2 осн. часа (из 5)
	Темы по выбору	1 час

Примечания:

Компьютерная графика обычно вызывает огромный интерес у студентов и потому естественно ожидать отличной мотивации, особенно, если в программе курса предусмотрена возможность создания графической системы. Хотя программная реализация является в данном курсе основной компонентой данного курса, необходимо также подчеркнуть математические основания данной предметной области.

Программные средства играют особо критичную роль в данном курсе. Хотя студентам будет полезно освоить базовые понятия на абстрактном уровне, им также необходимо опробовать сложные графические библиотеки, которые смогут неимоверно расширить возможности студентов по созданию интересных приложений. Помимо графических API, ориентированных на программирование, в курсе можно рассмотреть и другие пакеты, такие как средства для работы с мультимедиа, языки моделирования, виртуальную реальность.

Студенты, успешно окончившие данный курс, должны уметь:

- Анализировать графические и мультимедийные интерфейсы с точки зрения взаимодействия человека и компьютера.
- Применять основополагающие принципы разработки графических и мультимедийных систем.
- Описывать набор программных средств, которые могут быть использованы в процессе разработки графических и мультимедийных систем.
- Использовать существующие графические и мультимедийные пакеты для разработки удобных графических приложений.

$CS260_{\{S,T\}}$. Искусственный интеллект

Данный курс знакомит студентов с основными понятиями и приемами искусственного интеллекта.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Основные вопросы интеллектуальных систем: история искусственного интеллекта; философские вопросы; основные определения; моделирование мира; роль эвристик
- Поиск решений: пространства решений; метод "грубой силы"; поиск по первому совпадению; теория игр; поиск решений
- Представление знаний: обзор логики высказываний и логики предикатов; метод резолюций и доказательство теорем; нестрогий вывод; вероятностные доказательства; теорема Байеса
- Расширенный поиск: генетические алгоритмы; алгоритмы модельной закалки; локальный поиск
- Представление знаний и вывод: структурное представление; немонотонный вывод; вывод по действию или изменению; временной вывод и пространственный вывод; недостоверность; представление знаний для обнаружения ошибок, качественное представление
- Агенты: определение агентов; успешные применения и современные агентные системы; программные агенты, персональные помощники и доступ к информации; мультиагентные системы
- Машинное обучение и нейронные сети: определение и примеры машинного обучения; контролируемое обучение; автономное обучение; возобновляемое обучение; знакомство с нейронными сетями

- Системы планирования с искусственным интеллектом: планирование как поиск; планирование с участием оператора; пропозициональное планирование

Затрагиваемые разделы:

IS1	Основы интеллектуальных систем	1 осн. час
IS2	Поиск решений	5 осн. часов
IS3	Представление знаний	4 осн. часа
IS4	Расширенный поиск	6 часов
IS5	Расширенное представление знаний	5 часов
IS6	Агенты	3 часа
IS8	Машинное обучение и нейронные сети	5 часов
IS9	Системы планирования с искуственным интеллектом	5 часов
	Темы по выбору	6 часов

$CS261_W$. Искусственный интеллект и информация

Данный курс знакомит студентов с основами искусственного интеллекта и управления информацией.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Основные вопросы интеллектуальных систем: история искусственного интеллекта; философские вопросы; основные определения; моделирование мира; роль эвристик
- Поиск решений: предметные области; поиск методом "грубой силы"; поиск по первому совпадению; теория игр; поиск решений
- Представление знаний: обзор логики высказываний и логики предикатов; метод резолюций и доказательство теорем; нестрогий вывод; вероятностные доказательства; теорема Байеса
- Расширенный поиск: генетические алгоритмы; алгоритмы модельной закалки; локальный поиск
- Машинное обучение и нейронные сети: определение и примеры машинного обучения; контролируемое обучение; использование деревьев решения в обучении; обучаемые нейронные сети; обучаемые доверительные сети; алгоритм ближайшего соседа; теория обучения; проблема сверхпригодности; автономное обучение; возобновляемое обучение
- Информационные модели и системы: история и причины возникновения информационных систем; хранение и поиск информации; приложения, управляющие информацией; сбор и представление информации; анализ и индексирование; поиск, получение, связывание и навигация; конфиденциальность, целостность, безопасность и сохранность информации; масштабируемость, производительность и эффективность
- Системы баз данных: история и причины возникновения систем баз данных; компоненты баз данных; функциональность СУБД; архитектура базы данных и независимость данных
- Моделирование данных: моделирование данных; концептуальные модели; объектно-ориентированная модель; реляционная модель данных
- Реляционные базы данных: отображение концептуальной схемы в реляционную схему; целостность сущностей-объектов и ссылочная целостность; реляционная алгебра и реляционное исчисление
- Языки запросов к базам данных: обзор языков баз данных; SQL; оптимизация запросов; QBE и окружения 4-го поколения; встраивание непроцедурных запросов в процедурный язык; введение в объектно-ориентированный язык запросов (OQL)

Затрагиваемые разделы:

IS1	Основы интеллектуальных систем	1 осн. час
IS2	Поиск решений	5 осн. часов
IS3	Представление знаний	4 осн. часа
IS4	Расширенный поиск	3 часа
IS8	Машинное обучение и нейронные сети	3 часа
IM1	Информационные модели и системы	3 осн. часа
IM2	СУБД	3 осн. часа
IM3	Моделирование данных	4 осн. часа
IM4	Реляционные базы данных	3 часа
IM5	Языки запросов к базам данных	3 часа
SP6	Интеллектуальная собственность	1 осн. час (из 3)
	Темы по выбору	7 часов

$CS262_C$. Управление информацией и знаниями

Данный курс рассматривает информацию как универсальную идею, возникающую при рассмотрении целого ряда вопросов информатики, включая системы управления базами данных, искусственный интеллект, взаимодействие человека и компьютера, системы мультимедиа и телекоммуникации.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Информационные модели и системы: история и причины возникновения информационных систем; хранение и поиск информации; средства управления информацией; сбор и представление информации; анализ и индексирование; поиск, получение, связывание и навигация; конфиденциальность, целостность, безопасность и сохранение информации; масштабируемость, производительность и эффективность
- Системы баз данных: история и причины возникновения систем баз данных; компоненты баз данных; функциональность СУБД; архитектура базы данных и независимость данных; использование языка запросов к базе данных
- Моделирование данных: концептуальные модели; объектно-ориентированная модель; реляционная модель данных
- Реляционные базы данных: отображение концептуальной схемы в реляционную схему; целостность сущностей-объектов и ссылочная целостность; реляционная алгебра и реляционное исчисление
- Поиск решений: предметные области; поиск методом "грубой силы"; поиск по первому совпадению; теория игр; поиск решений
- Представление знаний: обзор логики высказываний и логики предикатов; метод резолюций и доказательство теорем; нестрогий вывод; вероятностные доказательства; теорема Байеса
- Основы человеко-машинного взаимодействия: мотивация; контексты взаимодействия человека и компьютера; создание и оценка эргономичных систем; модели поведения человека; учет человеческого разнообразия; принципы хорошего дизайна; технические ограничения; основы тестирования эргономичности ПО
- Основные вопросы интеллектуальных систем: история искусственного интеллекта; философские вопросы; основные определения; моделирование мира; роль эвристик
- Криптографические алгоритмы: исторический обзор криптографических алгоритмов; криптография с секретным ключом и проблема обмена ключами; криптография с открытым ключом; цифровые подписи; протоколы защиты
- Знакомство со сжатием данных: алгоритмы кодирования и декодирования; сжатие с потерями и сжатие без потерь
- Мультимедийная информация и мультимедийные системы
- Интеллектуальная собственность: основы интеллектуальной собственности; авторское право, патенты и коммерческая тайна; компьютерное пиратство; патенты на программное обеспечение; международное авторское право
- Конфиденциальность и гражданские свободы: этические и правовые основания защиты конфиденциальности (privacy); правовые особенности больших баз данных; технологии защиты конфиденциальности; свобода выражения в киберпространстве; международные и межкультурные последствия

Затрагиваемые разделы:

AL9	Алгоритмы криптографии	3 часа
NC7	Сжатие и декодирование данных	2 часа
HC1	Основы взаимодействия человека и машины	4 осн. часа (из 6)
IS1	Основы интеллектуальных систем	1 осн. час
IS2	Поиск решений	5 осн. часов
IS3	Представление знаний	4 осн. часа
IM1	Информационные модели и системы	3 осн. часа
IM2	СУБД	3 осн. часа
IM3	Моделирование данных	4 осн. часа
IM4	Реляционные базы данных	4 часа
IM13	Мультимедийная информация и системы	2 часа
SP6	Интеллектуальная собственность	3 осн. часа
SP7	Конфиденциальность и гражданские свободы	2 осн. часа

Примечания:

Учитывая, что данный курс освещает целый ряд предметов таких, как базы данных, искусственный интеллект и человеко-машинное взаимодействие, маловероятно, что курсы, подобные $CS262_C$, имеются в существующих учебных программах. Однако мы полагаем, что подобные курсы, предлагающие общий подход к различным темам, позволяют создать костяк общих знаний, а не просто разрозненных предметов. В этом случае общее знание заключается в методах управления, представления и манипулирования информацией. Оно включает, например,

вопросы хранения, поиска, кодирования и управления информацией безотносительно того, рассматриваются ли базы данных, интеллектуальные системы, телекоммуникации или графика. Также вопросы управления информацией включают социальные и этические аспекты, такие, как владение интеллектуальной собственностью и индивидуальные права на конфиденциальность (privacy).

Курс $CS262_C$ позволяет уделить больше времени изложению дополнительного материала, чем курсы, использующие сжатый подход. Например, в данный курс включены такие темы, как криптография, сжатие данных и мультимедиа. В зависимости от конкретных приоритетов преподавательского состава и интересов студентов, возможно включение в материал курса и других тем.

$CS270_T$. Базы данных

Данный курс знаком студентами с понятиями и методами работы баз данных.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Информационные модели и системы: история и причины возникновения информационных систем; хранение и поиск информации; приложения, управляющие информацией; сбор и представление информации; анализ и индексирование; поиск, получение, связывание и навигация; конфиденциальность, целостность, безопасность и сохранение информации; масштабируемость, производительность и эффективность
- Системы баз данных: история и причины возникновения систем баз данных; компоненты базы данных; функциональность СУБД; архитектура базы данных и независимость данных
- Моделирование данных: моделирование данных; концептуальные модели; объектно-ориентированная модель; реляционная модель данных
- Реляционные базы данных: отображение концептуальной схемы в реляционную схему; целостность сущностей-объектов и ссылочная целостность; реляционная алгебра и реляционное исчисление
- Языки запросов к базам данных: обзор языков баз данных; SQL; оптимизация запросов; окружения 4-го поколения; встраивание непроцедурных запросов в процедурный язык; введение в объектно-ориентированный язык запросов (OQL)
- Структура реляционных баз данных: структура баз данных; функциональные зависимости; нормальные формы; многозначные зависимости; зависимость соединения; теория представления данных
- Обработка транзакций: транзакции; неудачи и восстановление; управление параллелизмом
- Распределенные базы данных: распределенное хранение данных; обработка распределенных запросов; распределенная модель транзакций; управление параллелизмом; однородные и гетерогенные решения; клиент-серверная архитектура
- Физическое устройство баз данных: структура файлов; индексированные файлы; файлы с хэшированным доступом; файлы сигнатур; Б-деревья; файлы с плотным индексом; файлы с записями переменной длины; производительность базы данных и настройка

Затрагиваемые разделы:

HC1	Основы взаимодействия человека и машины	2 осн. часа (из 6)
IM1	Информационные модели и системы	3 осн. часа
IM2	СУБД	3 осн. часа
IM3	Моделирование данных	4 осн. часа
IM4	Реляционные базы данных	5 часов
IM5	Языки запросов к базам данных	4 часа
IM6	Устройство реляционных баз данных	4 часа
IM7	Обработка транзакций	3 часа
IM8	Распределенные базы данных	3 часа
IM9	Физическое устройство баз данных	3 часа
SP6	Интеллектуальная собственность	3 осн. часа
SP7	Конфиденциальность и гражданские свободы	2 осн. часа
	Темы по выбору	1 час

$CS270_S$. Управление информацией

Задача структуризации больших объемов информации потенциально различных видов представляется очень сложной. Обычно решение возникающих проблем зависит от выбора СУБД и типа сетевых коммуникаций. Данный курс обсуждает возникающие технические и социальные вопросы.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115), CS120.

Описание курса:

- Обзор управления информацией: история и причины возникновения информационных систем; общие проблемы управления информацией; деловая перспектива
- Социальные вопросы информационных технологий: интеллектуальная собственность; компьютерные преступления; право на частную жизнь; безопасность и гражданские свободы; необходимость в правовом и этическом окружениях; рекомендации по использованию компьютеров
- Знакомство с системами баз данных: история и причины возникновения систем баз данных; компоненты баз данных; функциональность СУБД; архитектура базы данных и независимость данных; использование языка запросов к базам данных; реляционная модель
- Создание баз данных: основополагающая методология; языки запросов к базам данных; специальные вопросы, связанные с базами данных
- Информационные системы специального назначения: локальные и глобальные сети; проблема поиска информации
- Структура и развитие информационных систем: устройство баз данных; реляционные базы данных; вопросы жизненного цикла
- Вопросы безопасности и контроля: обзор проблем и стандартных решений; целостность баз данных; транзакции; роль шифрования
- Оценка информационных систем

Затрагиваемые разделы:

IM1	Информационные модели и системы	2 осн. часа (из 3)
IM2	СУБД	2 осн. часа (из 3)
IM3	Моделирование данных	4 осн. часа
IM4	Реляционные базы данных	5 часов
IM5	Языки запросов к базам данных	5 часов
IM6	Устройство реляционных баз данных	2 часа
IM7	Обработка транзакций	3 часа
IM11	Хранение и поиск информации	2 часа
IM13	Мультимедийная информация и системы	2 часа
IM14	Цифровые библиотеки	2 часа
SP2	Социальные вопросы программирования	1 осн. час (из 3)
SP3	Методы и инструменты анализа	2 осн. часа
SP4	Профессиональная и этическая ответственность	2 осн. часа (из 3)
SP5	Риски и ответственность компьютерных систем	1 осн. час (из 2)
SP6	Интеллектуальная собственность	2 осн. часа (из 3)
SP7	Конфиденциальность и гражданские свободы	2 осн. часа
	Темы по выбору	1 час

Примечания:

Материал для этого курса основан на знаниях, полученных студентами на предыдущих курсах, особенно $CS140_S$. Данный курс обсуждает решения, необходимые для управления сложной информацией, а также методы хранения информации для упрощения поиска, сохраняющие естественную концептуальную структуру.

При развитии любой информационной структуры необходимо учитывать самые различные факторы. Одни из таких факторов является коммерческая или деловая перспектива. Соответственно, данный курс может быть использован как способ познакомить студентов с деловым миром и его различными аспектами, включая этические. Но в любом случае в курсе должен обсуждаться жизненный цикл программного продукта, включая фазы определения требований, спецификации, проектирования, реализации, тестирования и оценки. Понятия из областей телекоммуникации и взаимодействия человека и машины также должны рассматриваться в этом курсе – студентов необходимо ознакомить с этими идеями, чтобы привить им структурированный, продуманный подход к разработке подобных систем.

До некоторой степени, все информационные технологии зависят от технологий баз данных. Однако важен и ряд других принципов, таких как человеческий фактор и динамика WWW. В поисках удачных примеров можно знакомить студентов с подходящими web-сайтами и цифровыми библиотеками.

Обычно студенты правильно и ответственно реагируют на несчастные случаи и злоумышленные действия, связанные с компьютерами. Изучение подобных явлений может быть использовано, чтобы пробудить понимание важности изучения социальных и этических вопросов. Необходимо подойти к вопросу так, чтобы студенты понимали значимость этой темы.

В университетском кругу идеи данного курса могут быть освещены более ярко, если обратиться к другим правилам, которые должны соблюдать студенты. Эти правила демонстрируют жизненные примеры дисциплины, кото-

рая должна присутствовать в лабораториях, и практики управления, которая присутствует в деятельности обслуживающего персонала.

Студенты, успешно прошедшие данный курс, должны уметь:

- Описывать различные деловые и другие факторы (включая правовые и этические), влияющие на развитие информационных систем, что также включает требования удаленного доступа.
- Применять основные принципы технологии баз данных.
- Объяснять возможности распределенных информационных систем и проблемы, которые присутствуют в подобных системах.
- Различать общие механизмы обеспечения управления и безопасности, связанные с управлением информацией, и уметь эффективно применять эти механизмы.
- Объяснять необходимость кодексов поведения и правового окружения при использовании компьютеров.
- Приводить примеры нескольких приложений, которые порождают серьезные правовые и этические вопросы.

$CS280_T$. Социальные и профессиональные вопросы программирования

Данный курс знакомит студентов с социальными и профессиональными вопросами программирования.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112).

Описание курса:

- История программирования: предыстория – мир до 1946 года; история компьютерной аппаратуры, программного обеспечения, телекоммуникаций; пионеры программирования
- Социальный контекст программирования: знакомство с социальным эффектом программирования; социальное влияние телекоммуникаций; рост и управление Интернета; доступ к Интернету, включая половые аспекты; международные аспекты
- Методы и инструменты анализа: выдвижение и оценка этических соображений; проблема этического выбора; осознание социального контекста дизайна; определение неявных допущений и ценностей
- Профессиональная и этическая ответственность: общественные ценности и законы, по которым мы живем; природа профессионализма; различные формы профессиональной аттестации, их плюсы и минусы; роль профессионала в государственной политике; постоянное осознание последствий; моральные разногласия и их сигнализация; этические кодексы, поведение и практика; противодействие притеснениям и дискриминации; политики "допустимого использования" компьютеров на рабочем месте
- Риски и ответственность компьютерных систем: исторические примеры рисков при использовании программного обеспечения; последствия сложности ПО; оценка рисков и управление рисками
- Интеллектуальная собственность: основы интеллектуальной собственности; авторское право, патенты и коммерческая тайна; патенты на программное обеспечение; международные вопросы, связанные с интеллектуальной собственностью
- Конфиденциальность и гражданские свободы: этические и правовые основания защиты права на частную жизнь; правовые особенности больших баз данных; технологии защиты конфиденциальности; свобода выражения в киберпространстве; международные и межкультурные последствия
- Компьютерные преступления: история и примеры компьютерных преступлений; взлом ПО и его последствия; вирусы, черви и троянские кони; стратегии предотвращения преступлений
- Экономические вопросы программирования: монополии и их экономические последствия; влияние спроса и недостатка квалифицированного персонала на качество программных продуктов; стратегии установления цен в области программирования; различия в доступе к программным ресурсам и их возможные последствия
- Философские системы: философские системы, в частности утилитаризм и деонтологические теории; проблемы этического релятивизма; научная этика в исторической перспективе; различия в научных и философских подходах

Затрагиваемые разделы:

SP1	История информатики	1 осн. час
SP2	Социальные вопросы программирования	3 осн. часа
SP3	Методы и инструменты анализа	2 осн. часа
SP4	Профессиональная и этическая ответственность	3 осн. часа
SP5	Риски и ответственность компьютерных систем	2 осн. часа
SP6	Интеллектуальная собственность	3 осн. часа
SP7	Право на частную жизнь и гражданские свободы	2 осн. часа
SP8	Компьютерные преступления	3 часа
SP9	Экономические вопросы программирования	2 часа
SP10	Философские системы	2 часа
	Темы по выбору	17 часов

Примечания:

Учебная программа по информатике может включать социальные и профессиональные вопросы различным образом. Во многих отношениях, идеальны подходом было бы равномерное включение данного материала во многие курсы с тем, чтобы студенты имели возможность рассмотреть эти вопросы в контексте каждой конкретной технической области. К сожалению, эта методика не всегда дает ожидаемые результаты. Если преподавательский состав сознательно не уделяет серьезное внимание данной теме, то социальные и профессиональные вопросы получают низкий приоритет в контексте других курсов или вообще вытесняются из программы в связи с необходимостью изложить более традиционный материал.

Для того, чтобы обеспечить изложение этих вопросов, многие программы выделяют целый курс, посвященный социальным и профессиональным вопросам. При таком выборе необходимо добиться, чтобы материал был интересен для студентов, например, путем обсуждения этой темы в контексте конкретных примеров, существующих в информатике.

CS290$_T$. Разработка программного обеспечения

Данный курс является интенсивным и ориентированным на практическое применение введением в методы разработки программного обеспечения, используемые для создания диалоговых приложений средних размеров, использующих большие объектно-ориентированные библиотеки для создания удобного графического пользовательского интерфейса. Темы данного курса включают в себя программирование событийно-управляемых систем, компьютерную графику, взаимодействие человека и машины, а также графические пользовательские интерфейсы.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Описание курса:

- Программирование событийно-управляемых систем: методы обработки событий; распространение событий; управление параллельным исполнением при обработке событий; обработка исключений
- Использование программных интерфейсов приложений: программирование с использованием API; программы просмотра классов и подобные инструменты; программирование на примерах; отладка программы, использующей программный интерфейс приложения; программирование компонент
- Компьютерная графика: растровые и векторные графические системы; устройства отображения видеоинформации; физические и логические устройства ввода; вопросы, с которыми сталкивается разработчик графической системы
- Основы человеко-машинного взаимодействия: мотивация и контексты взаимодействия человека и компьютера; создание и оценка эргономичных систем; модели поведения человека; учет человеческого разнообразия; принципы хорошего дизайна; технические ограничения; основы тестирования эргономичности ПО
- Оценка эргономичности ПО: выделение целей тестирования; стратегии оценки
- Разработка эргономичных систем: подходы, характеристики и обзор процесса; методы и инструменты прототипирования
- Графические пользовательские интерфейсы: графические API; выбор стилей и методов взаимодействия; аспекты человеко-машинного взаимодействия в проектировании графики (внешний вид, цвета, шрифты, маркировка); управление геометрией; программные окружения для создания графических пользовательских интерфейсов
- Методы разработки ПО: объектно-ориентированные анализ и проектирование; проектирование компонент; требования и спецификации к ПО; прототипирование; характеристики легко сопровождаемого ПО; повторное использование программного обеспечения; управление командой разработчиков; планирование графика

Затрагиваемые разделы:

NC1	Введение в распределенные вычисления	2 осн. часа
NC2	Телекоммуникации и сети	2 осн. часа (из 7)
NC3	Сетевая безопасность	3 осн. часа
NC4	WWW как пример клиент-серверного программирования	3 осн. часа
PF5	Программирование событийно-управляемых систем	4 осн. часа
HC1	Основы взаимодействия человека и машины	6 осн. часов
HC2	Построение простого графического интерфейса	2 осн. часа
HC3	Оценка эргономичности ПО	1 час
HC4	Создание эргономичного ПО	1 час
HC5	Проектирование графического пользовательского интерфейса	3 часа
HC6	Программирование графического пользовательского интерфейса	3 часа

GV1	Основы методов программирования графики	2 осн. часа
GV2	Графические системы	1 осн. час
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	3 осн. часа (из 5)
SE3	Программные средства и окружения	2 осн. часа (из 3)
SE5	Спецификации и требования к ПО	2 осн. часа (из 4)
SE6	Проверка соответствия ПО	1 осн. час (из 3)
SE7	Эволюция программ	2 осн. часа (из 3)
SE8	Управление проектами	2 осн. часа (из 3)
	Темы по выбору	3 часа

CS291$_S$. Разработка программного обеспечения и системное программирование

Данный курс переносит идеи проектирования и разработки программного обеспечения, изложенные во вводной последовательности курсов, в контекст больших проектов. Темы данного курса включают в себя методы разработки больших программных проектов, углубленные вопросы объектно-ориентированного программирования, шаблоны проектирования, программирование клиент-серверных архитектур и принципы проектирования интерфейсов.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), CS210$_S$.

Описание курса:

- Создание больших систем: раздельная компиляция; вопросы проектирования; верификация и проверка правильности программ; интегрирование компонент; документирование
- Углубленные вопросы объектно-ориентированного программирования: модульность; управление хранением объектов; параллелизм; событийно-ориентированное программирование; общие шаблоны проектирования; повторное использование программного обеспечения
- Программирование клиент-серверных систем: программное обеспечение, необходимое для работы клиента и сервера; варианты устройства серверной части; стратегии разработки клиент-серверных архитектур; программные средства разработки клиент-серверных систем; связующее ПО
- WWW как пример клиент-серверного программирования: web-технологии; характеристики web-серверов; роль клиентских машин; понятие апплета; web-протоколы; программные средства для создания и управления web-сайтом; размещение информации и приложений; вопросы производительности
- Основы человеко-машинного взаимодействия: проектирование и оценивание эргономичных систем; соответствующие психологические и когнитивные теории; моделирование поведения пользователя; вопросы обучаемости и способности обучать
- Принципы проектирования человеко-машинного взаимодействия: создание диалоговых систем; основные методы проектирования интерфейса; примеры удачного и неудачного устройства интерфейса; программные средства и программные классы, поддерживающие разработку интерфейсов; метрики
- Принципы разработки графического пользовательского интерфейса: выбор стилей и методов взаимодействия; отражение аспектов человеко-машинного взаимодействия в стандартных графических элементах управления; аспекты человеко-машинного взаимодействия в разработке внешнего вида экрана; особые проблемы, связанные с цветами, звуком, видеоизображением и мультимедиа

Затрагиваемые разделы:

NC4	WWW как пример клиент-серверного программирования	2 осн. часа (из 3)
PF3	Основные структуры данных	6 осн. часов (из 14)
PF5	Программирование событийно-управляемых систем	4 осн. часа
HC1	Основы взаимодействия человека и машины	3 осн. часа (из 6)
HC3	Оценка эргономичности ПО	2 часа
HC4	Создание эргономичного ПО	2 часа
HC5	Проектирование графического пользовательского интерфейса	2 часа
HC6	Программирование графического пользовательского интерфейса	2 часа
PL6	Объектно-ориентированное программирование	4 осн. часа (из 10)
SP5	Риски и ответственность компьютерных систем	1 осн. час (из 2)
SE1	Проектирование ПО	2 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	3 осн. часа (из 5)
SE4	Процессы разработки ПО	1 осн. час (из 2)
SE5	Спецификации и требования к ПО	2 осн. часа (из 4)
SE6	Проверка соответствия ПО	2 осн. часа (из 3)
SE7	Эволюция программ	1 осн. час (из 3)
SE8	Управление проектами	1 осн. час (из 3)

Примечания:

Важным шагом в обучении хорошего разработчика программного обеспечения является переход от программирования-в-малом к программированию в больших программных проектах. Цель данного курса заключается в упрощении этого перехода путем предоставления студентам возможности создания больших программ в несколько четко определенных этапов. Данный курс рассматривает требования к программному продукту на каждом шаге вместе с различными вопросами обеспечения качества. В практической части курса студенты учатся правильно использовать в создании большой системы ряд возможностей, предоставляемых типичным объектно-ориентированным языком.

Переход от маленьких проектов к большим совсем непрост. Студентам необходимо иметь перед собой ряд примеров и образцов, которым они могли бы следовать. В предложенном нами варианте эти примеры берутся из распределенного программирования и проектирования пользовательских интерфейсов, однако, возможны и другие варианты. Например, подобные курсы могут использовать примеры из области электронной коммерции, корпоративного программного обеспечения и другие примеры сложных серверов приложений. В любом случае, необходимо подчеркнуть важность управления сложностью, демонстрируя, как большие задания могут быть разбиты на малые, которые, в свою очередь, могут быть реализованы с использованием соответствующих алгоритмов. При таком подходе студенты увидят связь с предыдущими курсами, в том числе с изучением алгоритмов и теории сложности.

С переходом к большим системам качество пользовательского интерфейса становится критическим, так как интерфейс сильно влияет на удобство использования ПО. Поэтому данный курс обращается к изучению принципов человеко-машинного взаимодействия. Человеко-машинное взаимодействие может рассматриваться как отдельный предмет или как частный вопрос программной инженерии, но имеет смысл использовать первый подход, так как основные идеи данной темы будут необходимы в различных контекстах. Например, понимание принципов человеко-машинного взаимодействия помогает осознать причины устройства и развития многих программных систем, включая web-сайты, мультимедийные системы и т.д.

В качестве удобной отправной точки изучения человеко-машинного взаимодействия, можно предложить студентам оценить положительные и отрицательные стороны интерфейсов различных типов. Однако в конечном итоге, студенты должны продемонстрировать свое понимание принципов человеко-машинного взаимодействия путем создания своего достаточно сложного интерфейса. Полезно также ознакомить студентов с современными программными средствами создания человеко-машинных интерфейсов, включая программные средства и библиотеки классов, специально предназначенные для создания интерфейсов. Практические занятия также стоит проводить с использованием языков проектирования и соответствующих программных средств.

Студенты, успешно прошедшие данный курс, должны уметь:

- Применять основные принципы создания прикладного программного обеспечения и пользовательских интерфейсов.
- Применять принципы разработки приложений (в том числе проектирование и разработку различных объектов) для создания крупного программного продукта, обосновывая проектные решения, принятые на каждом этапе, и учитывая требования к качеству ПО.
- Знать и уметь применять основные методы эффективного и высокопроизводительного создания больших программных систем.
- Знать возможности сетевого программирования, а также инструменты, как технические, так и программные, при помощи которых реализуются эти возможности.
- Применять принципы, связанные с проектированием и развитием приложений, предназначенных для работы во всемирной сети.
- Иметь представление о теориях, которые лежат в основе принципов разработки человеко-машинных интерфейсов.
- Систематично оценивать качество интерфейсов широкого спектра программных продуктов.

$CS292_{\{C,W\}}$. Разработка программного обеспечения и профессиональная практика

Материал данного курса сочетает в себе ряд тем, связанных с проектированием, реализацией и тестированием программных продуктов средних размеров, а также практику участия в команде разработчиков подобного продукта. В дополнение к материалу по программной инженерии, данный курс также рассматривает материал, связанный с профессиональной и моральной ответственностью программиста, связанной с разработкой ПО и вопросами человеко-машинного взаимодействия.

Требования к слушателям: $CS226_C$ и $CS262_C$, или $CS221_W$ и $CS250_W$.

Описание курса:

- Программирование событийно-управляемых систем: методы обработки событий; распространение событий; обработка исключений

- Основы человеко-машинного взаимодействия: проектирование и оценка эргономичных систем; модели поведения человека; учет человеческого разнообразия; принципы хорошего дизайна; технические ограничения; основы тестирования эргономичности ПО
- Использование программных интерфейсов приложений: программирование с использованием API; программы просмотра классов и другие подобные инструменты; программирование на примерах; отладка программы, использующей программный интерфейс приложения; знакомство с программированием компонент
- Создание простого графического пользовательского интерфейса: принципы устройства графического пользовательского интерфейса; инструментарий разработчика графического пользовательского интерфейса
- Графические системы: растровые и векторные графические системы; устройства отображения видеоинформации; физические и логические устройства ввода; вопросы, с которыми сталкивается разработчик графической системы
- Процессы разработки ПО: жизненный цикл программного продукта и модели процесса; модели оценки процесса; метрики процесса разработки ПО
- Спецификации и требования к ПО: выявление требований к ПО; методы анализа требований; функциональные и нефункциональные требования; прототипирование; основные понятия методов формальных спецификаций программ
- Проектирование ПО: основные понятия и принципы проектирования ПО; шаблоны проектирования; архитектура ПО; структурное проектирование; объектно-ориентированные анализ и проектирование; проектирование компонент; проектирование с целью повторного использования
- Проверка соответствия ПО: планирование проверки соответствия; основы тестирования, включая создания плана тестирования и генерацию тестовых пакетов; методы черного и белого ящика; модульное тестирование, тестирование взаимодействия компонентов системы, проверка соответствия и системное тестирование; объектно-ориентированное тестирование; проверки кода
- Эволюция ПО: сопровождение ПО; характеристики сопровождаемого ПО; реинжиниринг; унаследованные системы; повторное использование ПО
- Управление программными проектами: управление группой разработчиков; планирование графика проекта; методы оценивания ПО; анализ рисков; обеспечение качества ПО; управление конфигурацией ПО; инструменты управления проектами
- Социальный контекст программирования: знакомство с социальным эффектом программирования; социальное влияние телекоммуникаций; рост и управление Интернета; доступ к Интернету; половые аспекты; международные аспекты
- Методы и инструменты анализа: выдвижение и оценка этических соображений; проблема этического выбора; осознание социального контекста идеи; определение допущений и ценностей
- Профессиональная и этическая ответственность: общественные ценности и законы, по которым мы живем; природа профессионализма; различные формы профессиональной аттестации, их плюсы и минусы; роль профессионала в государственной политике; постоянное осознание последствий; моральные разногласия и их сигнализация; этические кодексы, руководства и практика; противодействие притеснениям и дискриминации; политики "допустимого использования" компьютера на рабочем месте
- Риски и ответственность компьютерных систем: исторические примеры рисков при использовании программного обеспечения; следствия сложности ПО; оценка рисков и управление рисками

Затрагиваемые разделы:

PF5	Программирование событийно-управляемых систем	2 осн. часа (из 4)
HC1	Основы взаимодействия человека и машины	2 осн. часа (из 6)
HC2	Построение простого графического интерфейса	2 осн. часа
GV1	Основы методов программирования графики	2 осн. часа
GV2	Графические системы	1 осн. час
SP2	Социальные вопросы программирования	3 осн. часа
SP3	Методы и инструменты анализа	2 осн. часа
SP4	Профессиональная и этическая ответственность	3 осн. часа
SP5	Риски и ответственность компьютерных систем	2 осн. часа
SE1	Проектирование ПО	4 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	3 осн. часа (из 5)
SE3	Программные средства и окружения	1 осн. час (из 3)
SE4	Процессы разработки ПО	2 осн. часа
SE5	Спецификации и требования к ПО	3 осн. часа (из 4)
SE6	Проверка соответствия ПО	2 осн. часа (из 3)
SE7	Эволюция программ	3 осн. часа
SE8	Управление проектами	3 осн. часа

Б.4. Углубленные курсы

Комитет CC2001 решил не включать в печатный вариант отчета полные описания углубленных курсов, если эти курсы не входят в состав хотя бы одного из стандартных вариантов преподавания, описанных в главе 8. Вместо этого, мы планируем создать web-страницы для этих курсов, которые будут доступны с главной страницы, посвященной CC2001 (http://www.computer.org/education/cc2001). Список предлагаемых нами дополнительных курсов показан на рисунке Б-4.

Рисунок Б-4. Углубленные курсы по дисциплинам.

Дискретные структуры (DS)
CS301. Комбинаторика
CS302. Вероятность и статистика
CS303. Теория кодирования и информации

Методы вычислений (CN)
CS304. Методы вычислений
CS305. Численный анализ
CS306. Исследование операций
CS307. Статистическое моделирование
CS308. Математическое программирование
CS309. Вычислительная биология

Алгоритмы и теория сложности (AL)
CS310. Анализ алгоритмов - 2
CS311. Теория языков и автоматов
CS312. Криптография
CS313. Геометрические алгоритмы
CS314. Параллельные алгоритмы

Архитектура и организация ЭВМ (AR)
CS320. Углубленные вопросы архитектуры компьютеров
CS321. Параллельные архитектуры
CS322. Однокристальные системы
CS323. Разработка VLSI
CS324. Кодизайн

Операционные системы (OS)
CS325. Операционные системы (2)
CS326. Параллельные и распределенные системы
CS327. Надежные вычисления
CS328. Отказоустойчивость
CS329. Системы реального времени

Распределенные вычисления (NC)
CS330. Углубленные вопросы компьютерных сетей
CS331. Распределенные вычисления
CS332. Программирование мобильных устройств
CS333. Кластерное программирование
CS334. Сжатие данных
CS335. Управление сетями
CS336. Сетевая безопасность
CS337. Корпоративные сети

Языки программирования (PL)
CS340. Создание компиляторов
CS341. Проектирование языков программирования
CS342. Семантики языков программирования
CS343. Парадигмы программирования
CS344. Функциональное программирование
CS345. Логическое программирование
CS346. Языки сценариев

Взаимодействие человека и машины (HC)
CS350. Проектирование и оценка эргономичных структур
CS351. Графические пользовательские интерфейсы
CS352. Разработка мультимедийных систем
CS353. Разработка диалоговых систем
CS354. Использование компьютеров в коллективе

Компьютерная графика и визуализация (GV)
CS355. Компьютерная графика (2)
CS356. Компьютерная мультипликация
CS357. Визуализация
CS358. Виртуальная реальность
CS359. Генетические алгоритмы

Интеллектуальные системы (IS)
CS360. Интеллектуальные системы
CS361. Автоматическое доказательство теорем
CS362. Системы с базами знаний
CS363. Обучение машины
CS364. Системы планирования
CS365. Обработка естественного языка
CS366. Агенты
CS367. Робототехника
CS368. Символьные вычисления
CS369. Генетические алгоритмы

Управление информацией (IM)
CS370. Базы данных (2)
CS371. Устройство баз данных
CS372. Обработка транзакций
CS373. Распределенные и объектные базы данных
CS374. Информационная проходка
CS375. Организация информационных хранилищ
CS376. Мультимедийные информационные системы
CS377. Электронные библиотеки

Социальные и профессиональные вопросы (SP)
CS380. Профессиональное программирование
CS381. Социальный контекст программирования
CS382. Компьютеры и мораль
CS383. Экономические вопросы программирования
CS384. Правовые вопросы, связанные с компьютерами
CS385. Интеллектуальная собственность
CS386. Право на частную жизнь и гражданские свободы

Программная инженерия (SE)
CS390. Разработка ПО (2)
CS391. Программная инженерия
CS392. Проектирование ПО
CS393. Проектирование ПО и формальные спецификации
CS394. Практическая программная инженерия
CS395. Улучшение процессов разработки ПО
CS396. Компонентное программирование
CS397. Программные окружения
CS398. Системы с повышенными требованиями к надежности

Б.5. Курсовые и дипломные работы

Как уже говорилось в разделе 9.3, мы считаем, что студентам в процессе обучения необходимо выполнить существенный групповой проект. В некоторых случаях такой опыт может быть приобретен в рамках обычных курсов, например, по программной инженерии, но все-таки более распространенным подходом является выполнение студентами крупной самостоятельной работы, требующей от студентов применения целого ряда изученных понятий и полученных навыков.

Учебная программа, изложенная в данном отчете, предлагает две возможные реализации студенческого проекта:

CS490. Дипломная работа,

выполняемая за один семестр или двухсеместровая последовательность:

CS490. Дипломная работа - 1
CS490. Дипломная работа - 2,

позволяющая студентам выполнить более интересный проект в течение одного учебного года.

Структура подобных курсов будет значительно варьироваться в различных учебных учреждениях. В некоторых программах дипломный проект может включать в себя дополнительные лекции, особенно, если более ранние курсы не излагают необходимые части материала. В любом случае, мы думаем, что любой учебный проект будет использовать материал совокупности знаний по информатике, например, как показано ниже:

HC1	Основы взаимодействия человека и машины	2 осн. часа (из 6)
HC5	Проектирование графического пользовательского интерфейса	2 часа
HC6	Программирование графического пользовательского интерфейса	2 часа
SE1	Проектирование ПО	4 осн. часа (из 8)
SE2	Использование программных интерфейсов приложения	3 осн. часа (из 5)
SE3	Программные средства и окружения	3 осн. часа
SE4	Процессы разработки ПО	2 осн. часа
SE5	Спецификации и требования к ПО	2 осн. часа (из 4)
SE6	Проверка соответствия ПО	3 осн. часа
SE7	Эволюция программ	2 осн. часа (из 3)
SE8	Управление проектами	3 осн. часа
	Управление группой разработчиков	2 часа
	Умение работать в коллективе	2 часа

Независимо от того, рассматриваются ли данные темы в соответствующих лекциях или приобретаются по ходу выполнения проекта, задачей студенческого проекта должно быть практическое применение и закрепление пройденного теоретического материала.

КОМПЬЮТЕРНОЕ ОБЩЕСТВО ИНСТИТУТА ИНЖЕНЕРОВ ПО ЭЛЕКТРОТЕХНИКЕ И ЭЛЕКТРОНИКЕ (IEEE-CS)

Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720 USA
http://computer.org; Phone: +1 714 821 8380; Fax: +1 714 821 4641

Цели организации: Компьютерное общество IEEE было создано для развития и усовершенствования теории и практики компьютерных и информационных технологий. Объединяя порядка 100 000 членов, общество является мировым лидером среди организаций компьютерных профессионалов. С момента своего основания в 1946 году компьютерное общество стало самым большим техническим обществом в IEEE.

Деятельность: Компьютерное общество реализует свою миссию путем организации конференций, изданий, технических комитетов, групп по подготовке стандартов и локальных студенческих отделений. Оно спонсирует (полностью или частично) более 140 ежегодных конференций, симпозиумов и совещаний по всему спектру тем информатики.

Структура организации: Более 100 000 представителей индустрии, академических институтов и государственных организаций всего мира являются членами Компьютерного общества. Добровольцы из членов общества могут участвовать в различных коллегиях, комитетах и рабочих группах IEEE-CS.

Руководство: Исполнительный директор, д-р David W. Hennage, имеет более чем 30-летний опыт управления и руководства ассоциаций в инженерной, научной и образовательной областях. Президент, д-р Willis K. King, принимал участие в деятельности Компьютерного общества в течении более чем 20 лет. Доктор King с 1969 года работает на факультете информатики в университете города Хьюстона.

Публикации: Компьютерное общество IEEE публикует, распространяет и сохраняет для потомков более 20 периодических изданий, предлагающих читателям рецензируемые статьи и исследовательские работы во всех областях информатики, включая искусственный интеллект, компьютерную аппаратуру, графику, сетевые технологии, информационные технологии, разработка ПО, мультимедиа и т.д. Неоднократно завоевывавший различные награды журнал *Computer* доставляется каждому члену Общества. *Computer* – это ежемесячное издание, которое держит членов IEEE-CS в курсе последних технологических новостей, тенденций и других вопросов, важных для профессионалов.

Цифровая библиотека: Цифровая библиотека Компьютерного общества является объемной электронной коллекцией, включающей выпуски 18 периодических изданий Общества (опубликованные после 1988 года), а также труды более 850 различных конференций (опубликованные после 1995 года). 68 000 хранимых в ней статей и работ делают Цифровую библиотеку ценным исследовательским инструментом для профессионалов в любой области информатики.

Технические комитеты: Технические комитеты общества – это глобальные коллективы профессионалов с общими интересами в конкретной области информатики. Члены этих коллективов взаимодействуют через электронные средства общения, встречаются на конференциях и т.д. Их размер варьируется от 500 до 10 000 членов. Существует 41 технический комитет со специализациями в таких областях как архитектура ЭВМ, операционные системы, Интернет, программная инженерия, безопасность и др. Большинство технических комитетов издают и бесплатно распространяют для своих членов информационные бюллетени.

Дистанционное обучение: В рамках своей программы дистанционного обучения Компьютерное общество предлагает своим членам 100 учебных курсов, доступных через Интернет. Покрываемые темы включают Java, управление проектами, Cisco, HTML, Unix, CompTIA, безопасность Windows в сетях и др. Все курсы сертифицированы производителями и многие фокусируются на помощи членам общества в подготовке к сертификационным экзаменам. Эта услуга требует как минимум 56K Internet соединения и является бесплатной для членов общества.

Рабочие группы по стандартизации: IEEE-CS является лидером компьютерной индустрии в разработке широко применимых, технически совершенных стандартов. Существует более 200 рабочих групп, относящихся к 11 комитетам по стандартизации IEEE-CS. Мы постоянно приглашаем членов общества принимать участие в разработке стандартов, и тысячи профессионалов откликаются на этот призыв.

Награды: Чтобы поощрять членов общества, достигших выдающихся результатов, IEEE-CS спонсирует действенную и престижную программу наград. Эти награды отмечают как технические достижения, так и служение профессии и обществу.

АССОЦИАЦИЯ ПО ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ (ACM)

1515 Broadway, New York, NY 10036 USA; http://www.acm.org;
1.800.342.6626 (USA and Canada) or +212.626.0500 (Global)

Цели организации: ACM является международной научной и образовательной организацией, развивающей информатику как искусство, науку и прикладную дисциплину. Содействуя открытому обмену информацией и пропагандируя профессиональные и этические ценности, ACM служит как общественным, так и профессиональным интересам. Основанная в 1947 году, Ассоциация по вычислительной технике является старейшим и крупнейшим в мире образовательным и научным сообществом профессионалов компьютерной науки. Она способствует постоянной информированности своих членов о новых тенденциях, направлениях и событиях в информатике и новых технологиях.

Деятельность: Для достижения своих целей, ACM организует конференции, публикации, образовательные программы, общественные слушания и специализированные тематические группы. Ежегодно ACM спонсирует порядка 100 конференций, включая конференции по компьютерам и гражданским свободам (CFP), компьютерной графике (SIGGRAPH) и объектно-ориентированным программным системам, языкам и приложениям (OOPSLA).

Структура организации: Более 75 000 представителей индустрии, академических институтов и государственных организаций всего мира являются членами ACM. Добровольцы общества могут участвовать в различных коллегиях, комитетах и рабочих группах ACM, в том числе, в руководящих органах Ассоциации.

Руководство: Исполнительный директор, д-р John R. White, ранее работал менеджером лаборатории информатики в Xerox PARC (Palo Alto Research Center). Президент, д-р Stephen R. Bourne, предприниматель из El Dorado Ventures, автор командного языка UNIX, известного как "Bourne Shell". Он осуществлял техническое руководство различных подразделений крупнейших технологических компаний, включая Sun Microsystems, Digital Equipment, Silicon Graphics и Cisco Systems. Вице-президент Maria Klawe является профессором и деканом факультета информатики в University of British Columbia. Д-р David Wise, профессор информатики из Indiana University, является Секретарем-Казначеем.

Публикации: ACM публикует, распространяет и сохраняет для потомков около 25 периодических изданий, описывающих оригинальные исследования ведущих разработчиков в области компьютерных и информационных технологий. Редакционная коллегия *Communications of the ACM*, нашего главного издания, задействует в своей работе ряд наиболее известных и уважаемых ученых. Другие издания включают *Crossroads*, студенческий журнал ACM, *Interactions*, посвященный взаимодействию человека и компьютера, и *Intelligence*, посвященный вопросам искусственного интеллекта. Новые издания ACM, являющиеся Интернет-изданиями (см. www.acm.org), включают *ACM TECHNews*, новостной дайджест для занятых ИТ-профессионалов, *Ubiquity*, журнал для критического и глубокого анализа проблем ИТ, и *e-Learn*, Интернет-журнал по дистанционному обучению.

Портал ACM: Портал ACM служит входом в цифровую библиотеку ACM и путеводитель ACM. Цифровая библиотека ACM включает обширную электронную коллекцию из более, чем 20 ACM изданий, включая более чем 40-летний архив журналов и трудов конференций ACM. Путеводитель ACM предоставляет доступ к громадным библиографическим ресурсам из компьютерных книг, журналов, сборников докладов и тезисов. Портал ACM также позволяет пользователям создавать их собственные "кабинеты" в библиотеке, что дает возможность организовывать, сохранять и совместно использовать статьи и библиографии по интересующим пользователей темам.

Тематические Группы: 34 тематические группы (Special Interest Groups, SIGs) ACM работают над различными проблемами ИТ-индустрии, включая компьютерную графику, человеко-машинный интерфейс, искусственный интеллект, извлечение информации, мобильные коммуникации, языки программирования и сети. Каждая тематическая группа выбирает регламент своей работы, наиболее подходящий ее целям и задачам. Многие тематические группы спонсируют ведущие конференции и семинары, публикуют информационные бюллетени и другие издания и поддерживают работу электронных форумов.

Награды: Среди наград, ежегодно присуждаемых ACM, наиболее значительными являются A.M. Turing Award (известная как "нобелевская премия информатики"), Grace Murray Hopper Award для молодых профессионалов, Paris Kanellakis Theory and Practice Award, Karl V. Karlstrom Outstanding Educator Award и Allen Newell Award, отмечающая вклад в развитие интеграции информатики и других дисциплин.

Фирма ISD, созданная в 1993 г. для разработки прикладного программного обеспечения нового поколения, сегодня является одной из ведущих компаний Днепропетровска, работающих в этой отрасли. Основное направление: создание информационных систем для обслуживания больниц и медицинских лабораторий. Компания стабильно развивается и сейчас наш персонал насчитывает уже более 170 человек.

Наша деятельность – это разработка медицинских клиент-серверных приложений на Visual C++ и Java, с применением СУБД dbVista и ORACLE. Весь документооборот ведется на английском языке с использованием UML.

ISD тесно сотрудничает с рядом фирм Европы и США, разрабатывающих аналогичное ПО для американских госпиталей и лабораторий.

В нашей компании постоянно открыты вакансии для программистов и тестеров программного обеспечения, владеющих английским языком. В конкурсе могут участвовать выпускники ВУЗов Украины (успешно прошедним интервью мы помогаем с переездом в Днепропетровск) и специалисты с различным уровнем подготовки. Наши предложения будут соответствовать Вашему уровню. Преимущество имеют сертифицированные (MCSD, OCP) специалисты. Диапазон вакансий – широкий. Обращаться ежедневно с 10:00 до 17:00 по тел. (056) 778-26-68/67. Наш адрес в Интернете: http://www.isd.dp.ua.

Спонсор проекта:
ЛАНИТ-ТЕРКОМ
Исследования и разработки

20 лет наукоемкого производства в Петергофе

ЛАНИТ-ТЕРКОМ имеет тесные связи с кафедрой системного программирования математико-механического факультета и НИИ Информационных Технологий Санкт-Петербургского государственного университета. Основателем и бессменным руководителем коллектива является доктор физико-математических наук, профессор Андрей Николаевич Терехов.

Наш коллектив в течение 20 лет ведет исследования и разработки в следующих областях:

- разработка оригинального наукоемкого системного программного обеспечения;
- адаптация (портирование) свободно распространяемого ПО для заказных процессоров;
- решения в области технологии разработки программного обеспечения и CASE средств;
- разработка вычислительных средств с оптимизированным встроенным вычислителем;
- реинжиниринг прикладного программного обеспечения;
- программное обеспечение телефонных станций;
- разработка оборудования цифровых АТС и модернизация существующих станций;
- мобильные технологии;
- компьютерная телефония.

Мы приглашаем все заинтересованные организации к сотрудничеству в учебных, исследовательских и коммерческих проектах.

Россия, 198904,
Санкт-Петербург
Университетский пр., 28

Тел/факс: +7(812)4287109
e-mail: info@tercom.ru
Интернет: www.tercom.ru

АП КИТ

Ассоциация Предприятий Компьютерных
и Информационных Технологий

Россия, Москва, (095) 267-3622, 267-3450; e-mail: info@apkit.ru; www.apkit.ru

АП КИТ – Ассоциация предприятий компьютерных и информационных технологий образована осенью 2001г. и по составу участников это самое представительное некоммерческое объединение ИТ-отрасли в России. Ее членами являются крупнейшие отечественные и мировые компании в области программного обеспечения, производства компьютеров и оборудования, ведущие отечественные дистрибуторы, системные интеграторы, российские производители и разработчики. Ассоциация призвана выражать консолидированное мнение ИТ-отрасли, отстаивать ее интересы. Это объединенная работа с государством и, в то же время, рабочий механизм взаимодействия внутри отрасли. Это согласованные действия, направленные на рост и развитие рынка.

Основные цели Ассоциации:

- защита и представление интересов компаний, работающих в области информационных технологий;
- влияние на выработку представительными и исполнительными органами власти РФ правовой, экономической и социальной политики, отвечающей профессиональным интересам членам Ассоциации, и содействие ее эффективной реализации;
- противодействие монополизму и недобросовестной конкуренции в области информационных технологий;
- содействие развитию компаний, работающих в области информационных технологий, построению цивилизованных рыночных отношений в России;
- содействие созданию благоприятных финансово-экономических условий для членов Ассоциации;
- представление интересов членов Ассоциации в международных организациях;
- координация предпринимательской деятельности своих членов, представление и защита их общих имущественных интересов.