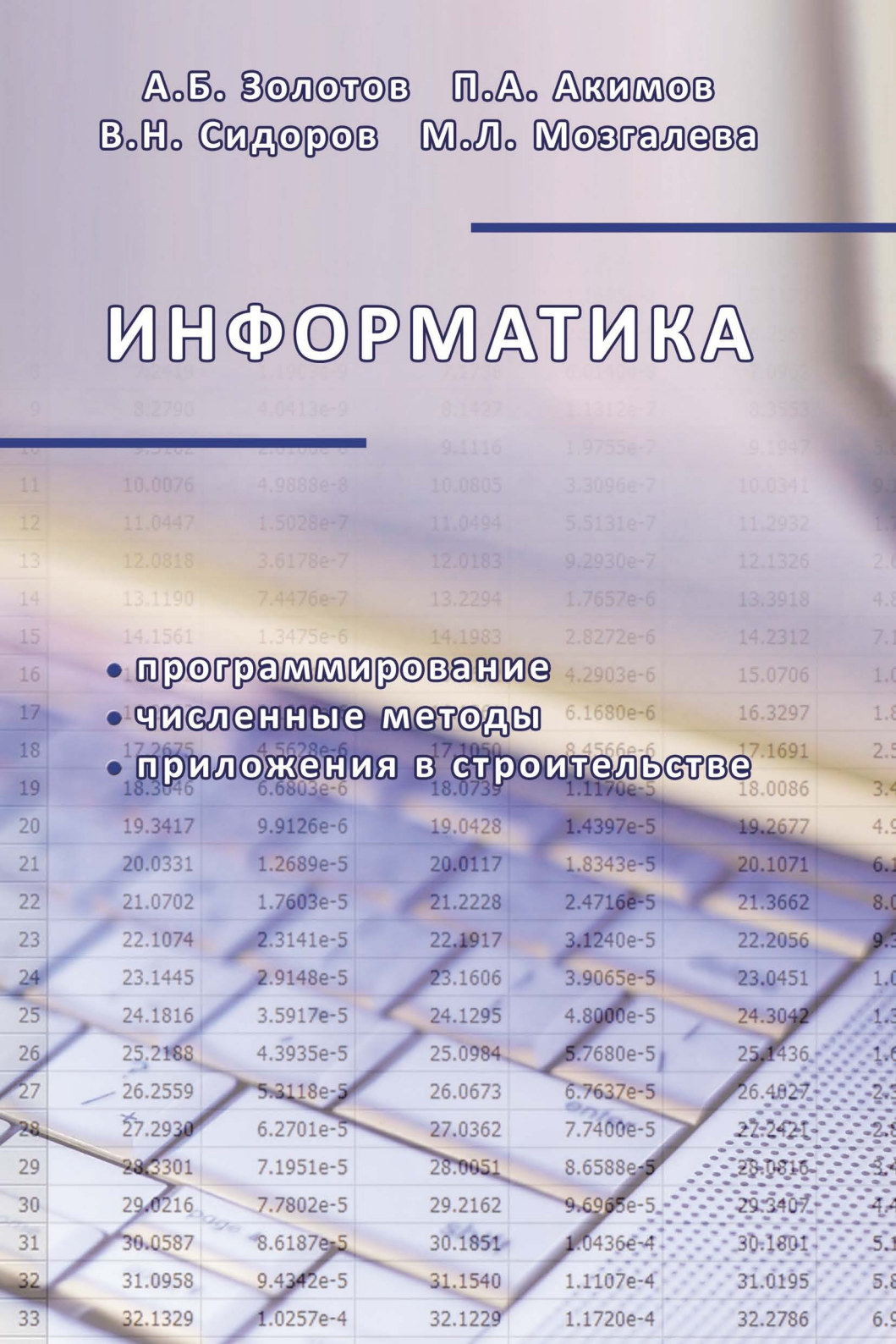
А.Б. Золотов П.А. Акимов
В.Н. Сидоров М.Л. Мозгалева

ИНФОРМАТИКА

- программирование
- численные методы
- приложения в строительстве



А.Б. Золотов, П.А. Акимов,
В.Н. Сидоров, М.Л. Мозгалева

ИНФОРМАТИКА

Рекомендовано
Учебно-методическим объединением вузов РФ
по образованию в области строительства
в качестве учебника для студентов,
обучающихся по направлению 270100 «Строительство»

Издательство АСВ
Москва
2010

УДК 519.6:539.3:624.04

Рецензенты:

Заведующий кафедрой строительной механики и вычислительных технологий
Пермского государственного технического университета
советник РААСН, профессор, доктор технических наук *Г.Г. Кашеварова*

Заведующий кафедрой «Инженерная и компьютерная графика»
Южно-Российского государственного технического университета
профессор, доктор технических наук *П.П. Гайджуров*

Золотов А.Б., Акимов П.А., Сидоров В.Н., Мозгалева М.Л.
Информатика. Учебник. – М.: Издательство АСВ, 2010. – 336 стр.

ISBN 978-5-93093-752-7

Содержание книги подразделено на пять глав, в которых имеется обзор и характеристика современных языков и систем программирования, рассматриваются вопросы программирования на языке Фортран современных стандартов, излагаются основы численных методов, описываются их приложения к решению конкретных прикладных задач, главным образом строительной направленности, приводятся сведения о математических библиотеках IMSL и SSP.

Учебник предназначен для студентов технических вузов, обучающихся по дисциплине «Информатика», частично может использоваться при изучении дисциплин «Строительная информатика», «Алгоритмические языки и программирование», «Численные методы», «Программные и аппаратные средства информатики», рекомендуется студентам старших курсов при выполнении курсовых и дипломных проектов и работ, а также магистрантам и аспирантам. Кроме того, издание полезно для самостоятельного изучения.

ISBN 978-5-93093-752-7

Учебник

Золотов Александр Борисович
Акимов Павел Алексеевич
Сидоров Владимир Николаевич
Мозгалева Марина Леонидовна

ИНФОРМАТИКА

Компьютерный набор и верстка: П.А. Акимов, В.Н. Сидоров, М.Л. Мозгалева.
Редактор: Г.М. Мубаракшина. Дизайн обложки: Н.С. Романова

Лицензия ЛР № 0716188 от 01.04.98. Подписано к печати 26.05.10. Формат 70*100/16
Бумага офс. Гарнитура Таймс. Печать офсетная.

Усл. 21,0 п.л. Тираж 1000 экз. Заказ №____

ООО «Издательство АСВ», 129337, Москва, Ярославское шоссе, 26, отдел реализации –
оф. 511, тел., факс: (499)183-56-83, e-mail: iasv@mgsu.ru, http://www.iasv.ru/

Предисловие

Прошло относительно немного времени с тех пор, как в школах и вузах страны начали изучать дисциплину «Информатика». В настоящее время эта дисциплины по праву является базовой в системе высшего образования и в комплексе с другими классическими дисциплинами призвана создавать фундамент профессионального образования в вузе.

Эта область знаний обширна и сегодня ее границы, а также содержание самой дисциплины «Информатика», трактуются по-разному. Одни считают ее велением времени, данью моде, явлением преходящим, или рассматривают информатику как некую премудрость. Другие воспринимают информатику как сугубо практическую дисциплину, учащую не знаниям, а умению «правильно нажимать кнопки» персонального компьютера, работая в наиболее распространенных программных средах, и осваивая лишь их офисные приложения. Надеемся, что читатель с иронией относится к такой трактовке. Достижение профессиональных компетенций при столь «радикальном» подходе к изучению этой дисциплины весьма сомнительно – будут приобретаться лишь навыки и немного умения, тогда как знания не появятся. Кроме того, учитывая высокую динамику смены одних программных продуктов и технических средств другими, вряд ли вообще когда-либо удастся сформировать адекватные современным реалиям умения и навыки.

Разные взгляды на область знаний, названную «Информатикой», и на ее границы определили и разные подходы к формированию содержания этого курса в вузах. В настоящее время существует несколько примерных программ по дисциплине «Информатика», и каждое учебное заведение, реализуя различные направления подготовки, проводит обучение в соответствии с потребностями соответствующих предметных областей.

Опыт ведущих отечественных и зарубежных образовательных центров позволяет сформулировать ряд принципов, которые должны быть основными при организации учебного процесса по дисциплине «Информатика»:

- активное использование математического аппарата, что предполагает изучение принципов математического моделирования и основ численных методов;
- реальное овладение навыками практической работы со средствами передачи и обработки информации, что требует выполнения многочисленных упражнений на программирование;
- изучение, прежде всего, универсальных теорий и методов, которые не зависят от используемой операционной системы и конкретной реализации и версии того или иного изучаемого языка программирования.

В таком понимании информатика формирует для будущих специалистов особую методику освоения информационных технологий и решения профессиональных задач, основанную «на развитии в широком применении методов математического моделирования и вычислительного эксперимента и служащую ближайшим стратегическим резервом ускорения научно-техни-

ческого прогресса» [55].

Предлагаемая вниманию читателя книга написана на основе курсов лекций по информатике, читаемых в последние годы студентам Московского государственного строительного университета (МГСУ) преподавателями кафедры информатики и прикладной математики. Основным разработчиком и идеологом преподавания курсов был Почетный член Российской академии архитектуры и строительных наук, Почетный работник высшего образования России, Почетный профессор МГСУ, профессор, доктор технических наук Александр Борисович Золотов (1937 – 2008). В формировании курса этой дисциплины также участвовал ряд ведущих преподавателей кафедры, среди них Л.А. Багиров, М.В. Белый, В.Н. Варапаев, С.Ю. Доброхотов, А.М. Купфер и другие.

За рамками данного учебника сознательно оставлено изложение вопросов, связанных с классификацией и устройством современных компьютеров, описанием компьютерных сетей, основами работы с популярными операционными системами, антивирусными пакетами, средствами защиты информации и офисными приложениями. Это было сделано с тех позиций, что все вышеперечисленное является основой школьных курсов дисциплины «Информатика».

Учебник разбит на пять глав, отражающих следующие аспекты информатики: обзор современных языков и систем программирования, основы программирования на языке Фортран, основы численных методов, численные методы решения инженерных задач и математические библиотеки численных методов.

Принятый авторами стиль изложения во многом определяется характером будущей работы студентов. Для них информатика и численное моделирование в первую очередь будут современным наукоемким инструментом решения прикладных технических проблем. Кроме того, рассматриваемый материал позволяет понять принципы построения и функционирования современных высокопроизводительных программных комплексов промышленного типа, распространенных в строительной отрасли, и основы реализуемых в них методов.

В ходе изложения мы не всегда стремились к максимальной общности и безупречной строгости формулировок. Представляется, что сделанные отступления от распространенного педантичного стиля современной литературы по численным методам будут способствовать большему пониманию и усваиванию излагаемого материала студентами технических высших учебных заведений.

Весь теоретический материал снабжен большим количеством примеров, которые носят как поясняющий, так и контрольный характер. Заметим, что задания на выполнение практических и лабораторных работ, представленные в Главе 4, носят модельный характер, удобный для отладки представленных программ, с легко предсказуемыми ответами.

В конце книги приведен библиографический список, пользуясь которым желающие могут более полно изучить рассмотренные вопросы.

Учебник предназначен для студентов технических высших учебных заведений, обучающихся по дисциплине «Информатика», частично может использоваться при изучении дисциплин «Строительная информатика», «Алгоритмические языки и программирование», «Программные и аппаратные средства информатики», рекомендуется студентам старших курсов при выполнении курсовых и дипломных проектов и работ, а также аспирантам. Кроме того, издание полезно для самостоятельного изучения.

Авторы благодарны рецензентам книги, профессорам Г.Г. Кашеваровой (Пермский государственный технический университет) и П.П. Гайджурову (Южно-Российский государственный технический университет (Новочеркасский политехнический институт)), за проявленный интерес и доброе отношение к работе.

Большую помощь в подготовке книги авторам оказала генеральный директор издательства Международной ассоциации строительных высших учебных заведений (Издательство АСВ), профессор Н.С. Никитина. Также благодарим сотрудников издательства АСВ за содействие в подготовке книги к изданию.

Авторы будут признательны за любые высказанные мнения и замечания по содержанию и изданию книги.

Издание настоящего учебника осуществлено в рамках выполнения следующих научно-исследовательских работ:

1. Грант № 09-08-13697 Российского фонда фундаментальных исследований «Разработка, исследование и развитие корректных численно-аналитических методов расчета строительных конструкций, зданий и сооружений регулярной структуры» на 2009-2010 гг.;

2. Проект «Разработка теории и алгоритмов построения корректных аналитических решателей многоточечных краевых задач применительно к расчетам строительных конструкций», выполняемый по Аналитической ведомственной целевой программе «Развитие научного потенциала высшей школы (2009-2010 годы)» (регистрационный номер: 2.1.2/6414);

3. Грант МД-4641.2009.8 Президента Российской Федерации для государственной поддержки научных исследований молодых российских ученых – докторов наук «Разработка и развитие корректных дискретно-континуальных методов статического и динамического расчета строительных конструкций, зданий и сооружений на основе построения точных аналитических решений многоточечных краевых задач строительной механики» на 2009-2010 гг.

Акимов Павел Алексеевич
e-mail: pavel.akimov@gmail.com

Сидоров Владимир Николаевич
e-mail: sidorov.vladimir@gmail.com

Мозгалева Марина Леонидовна
e-mail: marina.mozgaleva@gmail.com

Информатика – это фундаментальная естественная наука о структуре и общих свойствах информации, а также об эффективных правилах, методах и средствах сбора, обмена, хранения и обработки информации, рассматриваемой как отображение знаний и фактов, сведений, данных в различных областях человеческой деятельности.

Информация – это отображение в человеческом сознании знаний и фактов (сведений, данных), встречающихся и используемых в различных областях человеческой деятельности. При этом основными операциями, выполняемыми над информацией, являются *сбор, обмен, хранение и обработка*.

Сбор информации – это деятельность человека, часто с помощью специальных технических устройств, в ходе которой он получает необходимые сведения.

Обмен информацией – это процесс передачи информации на разные расстояния между различными объектами (между человеком и человеком, между человеком и техническим устройством, между различными техническими устройствами).

Хранение информации – это процесс поддержания информации в формате и на носителях, обеспечивающих ее использование в нужном виде и в нужное время, а также защита информации от ее несанкционированного использования.

Обработка информации – это процесс ее целесообразного преобразования.

Информационная технология – это какая-либо конкретная система средств, методов и способов поиска, приема, сбора, накопления, обработки и передачи информации.

Информатизация – это широкое внедрение современных информационных технологий в профессиональную деятельность специалистов различного профиля, в быт и досуг человека, в его учебную, научно-исследовательскую, управленческую, социальную деятельность.

Компьютеризация – это процесс оснащения организаций, предприятий и рабочих мест отдельных специалистов различными средствами вычислительной техники, объединения отдельных машин в компьютерные сети, установки и освоения современных компьютерных систем.

Модель – это материальный или идеальный образ некоторой совокупности реальных объектов или явлений, который при определенных обстоятельствах используется в качестве заменителя или представителя исходных объектов или явлений. Это образ, полученный с помощью концентрации внимания только на некоторых, важнейших с точки зрения решаемой задачи атрибутах рассматриваемых предметов или явлений и отбрасывания всех их несущественных свойств. При решении задач в различных областях дея-

тельности приходится строить различные модели. В информатике оперируют главным образом информационными и математическими моделями.

Информационная модель – модель объекта, явления, представленная информацией о нем, описывающей существенные для данного рассмотрения параметры и переменные характеристики объекта, связи между ними, входы и выходы объекта, и позволяющая путем подачи на модель информации об изменениях входных величин моделировать возможные состояния объекта. Очевидно, что один и тот же объект, одно и то же явление, рассматриваемые с различных точек зрения, с целью ответа на разные вопросы о нем, могут иметь различные информационные модели.

Математическая модель – это приближенное представление закономерности проявления некоторого класса объектов или явлений окружающего мира, выраженное в виде математических конструкций-аналогов и сформулированное в математических терминах и символах.

Математическое моделирование – это технология обработки информации об интересующих нас объектах, предусматривающая изучение и прогнозирование их проявлений с использованием возможностей математики. Процитируем здесь академика АН СССР А.Н. Самарского. «Сущность математического моделирования и его главное преимущество состоит в замене исходного объекта соответствующей математической моделью и в дальнейшем ее изучении (экспериментирование с нею) на ЭВМ с помощью вычислительно-логических алгоритмов. Математическое моделирование представляет собой естественное развитие и обобщение методов научного исследования, соединенных с современной информационной технологией. Цикл вычислительного эксперимента объект – модель – алгоритм – программа – ЭВМ – управление объектом отражает основные этапы процесса познания в нынешнем компьютерном воплощении. Здесь органично соединяются сильные стороны теоретических методов и натурного эксперимента. Работа с моделью, а не с объектом, оборачивается оперативным получением подробной и наглядной информации, вскрывающей его внутренние связи, качественные характеристики и количественные параметры. Многократно уменьшаются материальные и трудовые затраты, присущие традиционным экспериментальным подходам, дающим, как правило, лишь крупицы нужной информации. Вычислительный эксперимент не подвластен каким-либо ограничениям – математическая модель может быть безопасно испытана в любых мыслимых и немыслимых условиях» [55].

Алгоритм – это последовательность действий, которую необходимо выполнить над исходными данными, чтобы достичь поставленной цели на этапах сбора, хранения, обработки или передачи информации.

Алгоритм – это строгая конечная система правил, инструкций для исполнителя, определяющая некоторую последовательность конечного числа действий и после их выполнения приводящая к достижению поставленной цели.

Алгоритм – это всякая система вычислений, выполняемых по строго определенным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи.

Ключевыми понятиями информатики являются *информация*, *информационная модель*, *алгоритм*, *компьютер*.

Компьютер (*электронно-вычислительная машина* (*ЭВМ*)) – это электронное устройство, используемое для автоматизации процессов приема, хранения, обработки и передачи информации, которые осуществляются по заранее разработанным человеком алгоритмам (программам). Свое название компьютеры получили по своей основной функции – проведению вычислений. В настоящее время помимо вычислений компьютеры используются для обработки и управления информацией, оборудованием, а также для игр.

Алгоритмический язык – это язык записи алгоритмов, достаточно несложный, удобный для человека и «понятный» компьютеру.

Программа – это запись алгоритма в специальной, «понятной» компьютеру форме.

Данные – это информация, обрабатываемая программой, записанная в «понятной» компьютеру форме.

В последние десятилетия достижения человека во всех сферах его деятельности, в том числе в области науки и техники, обеспечиваются поиском, сбором, обменом, хранением и переработкой больших объемов информации.

Передача и переработка информации осуществляются, прежде всего, на основных «информационных магистралях», объединяющих ключевые этапы профессиональной деятельности человека. В строительстве такими этапами профессиональной деятельности являются изыскание, проектирование, производство строительных материалов, конструкций, воздвижение и оборудование объекта, его эксплуатация, реконструкция, утилизация. Не менее важны потоки информации на тех информационных коммуникациях, на которых обеспечивается успешное и разумное достижение целей строительства: соответствие требованиям устойчивого развития общества и экологии, государственное и правовое регулирование, инвестиции и страхование, экономика и организация труда.

У человечества есть мощный, удачный инструмент целенаправленной, организованной работы с большими потоками информации – это компьютер. Поэтому сегодня информатизацию общества справедливо отождествляют с использованием компьютерной техники [43,61].

Для организованного хранения и передачи информации разрабатываются целесообразные правила и универсальные форматы. Они должны быть удобными и обеспечивать доступность к любой информации в любой точке планеты и в любое время (конечно, кроме тех случаев, когда доступ к информации запрещен или ограничен законодательством государства или другими, как правило, производственными причинами). При этом в современных информационных технологиях уже не используются бумажные носители информации. На всех этих этапах работы с информацией используются не только технические, аппаратные средства, но и математические, в том числе алгоритмические конструкции. Но наиболее наукоемким, максимально ориентированным на математический арсенал является этап обработки (переработки) информации. Целенаправленную переработку информации об объекте отождествляют с процедурой формирования и «испытания» его математической модели, или с математическим моделированием. На этом этапе главным образом формулируются и решаются математические задачи. Академик АН СССР А.А. Самарский в свое время назвал этап переработки информации «интеллектуальным ядром» современных информационных технологий, в котором математическое моделирование играет роль наукоемкого фильтра, преобразующего «информационное сырье в готовый продукт, т.е. в точное знание» [56]. В свою очередь, переработка информации по технологии математического моделирования также неразрывно связана с использованием компьютеров. Компьютерное математическое моделирование, или *численное моделирование*, объектов и явлений – это вычислительный путь решения задачи обработки информации. Этот путь включает разработку и запись алгоритма, как правило, реализующего математические

численные методы, и решение задачи по этому алгоритму (испытание математической модели) компьютерными средствами.

Для формирования информационного образа знаний и фактов, сведений, данных об окружающей нас действительности создаются их информационные модели. При формировании информационной модели важно различать определяющие и несущественные характеристики объекта, окружающие его факторы, влияющие, слабо влияющие или вообще не влияющие на его конкретное рассматриваемое проявление.

Отвлечение от несущественных деталей принято называть *абстрагированием*. Абстрагирование является одним из важнейших инструментов при построении модели в какой-либо предметной области. Конечно, при абстрагировании осуществляется определенное огрубление картины реальной действительности. Однако концентрация внимания на наиболее важных аспектах, атрибутах позволяет выявить определенные свойства, закономерности и, следовательно, понять сущность изучаемого объекта, явления.

Адекватная модель – это модель, достаточно верно отражающая важнейшие особенности реальных объектов или явлений; она позволяет спрогнозировать поведение объекта в той или иной ситуации, описать процесс развития явления во времени, вовремя получить и использовать нужную информацию. Построение модели является наиболее важным, наиболее сложным и наиболее творческим этапом при изучении и передаче информации о любых объектах или явлениях. Разработка и использование модели с пользой требует объединенных усилий высококвалифицированных специалистов в конкретной предметной области и специалистов в области математики, прикладной математики и информатики.

Переработка информации об объекте подразумевает корректное формулирование и решение математической задачи так же с помощью компьютера. В основе формулировки этой задачи – математическая модель. Математическую модель следует воспринимать как эквивалент объекта, построенный на математическом языке, в математической форме, повторяющий основные закономерности, которым он подчиняется в своем проявлении, важнейшие его свойства, связи, присущие его взаимодействующим частям.

Как видно, понятие математической модели очень близко к понятию информационной модели. Часто специалисты рассматривают математическую модель как специфический, частный случай информационной модели.

Теперь остановимся на понятии *алгоритма*. В предыдущем разделе было приведено несколько его определений. Так или иначе, это то описание способа решения задачи, последовательности действий, которого следует придерживаться при работе с информацией для достижения цели. А собственно решение задачи заключается в исполнении (реализации) алгоритма.

Алгоритмы должны удовлетворять следующим свойствам: определенность или однозначность (исполнение одного и того же алгоритма в одних и тех же условиях различными исполнителями должно приводить к одинаковым результатам), понятность задания алгоритма его исполнителю, возможность исполнения алгоритма в тех или иных конкретных условиях, принципиальная достижимость результата.

Построение алгоритма представляет весьма сложный творческий процесс, на который иногда затрачивается длительное время. Искушенные люди отождествляют создание алгоритмов с искусством, различают и ценят удачные алгоритмические приемы и разработки.

Способы задания алгоритмов весьма разнообразны – словесный, графический (например, с помощью *блок-схемы* – нарисованной по специальным правилам схемы выполнения какой-либо последовательности действий) и другие.

Важным средством записи алгоритмов являются алгоритмические языки. Дело в том, что компьютер воспринимает и исполняет только алгоритмы, заданные в виде свойственных ему двоичных, машинных кодов. Однако этот естественный для компьютеров, обладающий всеми необходимыми свойствами способ задания алгоритмов, неоправданно очень сложен для использования человеком, не являющимся профессионалом в области компьютерной техники. Поэтому в информатике применяется ряд специальных способов, языков записи алгоритмов, которые, во-первых, призваны обеспечить соответствие алгоритма всем необходимым требованиям, а во-вторых, приспособить их для удобного использования, как человеком, так и после специальной автоматизированной обработки компьютером. Такие искусственные языки, использующиеся для записи алгоритмов и обеспечивающие им наличие всех необходимых свойств, называются *алгоритмическими языками*.

Запись алгоритма на одном из алгоритмических языков называется *компьютерной программой.*

СОВРЕМЕННЫЕ ЯЗЫКИ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ

§ 1.1. О роли и необходимости изучения языков программирования

Развитие научно-технического потенциала нашей страны напрямую связано с ростом потребности в технических и математических расчетах, а также в соответствующих программных продуктах. Популярные в России готовые пакеты математических программ (например, MathCAD, Mathematica, Maple, Matlab [22]) не решают проблемы, прежде всего, в связи с тем, что ориентированы главным образом на решение отдельных математических задач стандартными методами. Какой бы мощной ни была подобная математическая система, она не может решать любые задачи хотя бы потому, что постоянно возникают новые. Практические расчеты требуют творческого комплексного подхода, реализуемого лишь в классическом программировании. Заметим, что мощные математические системы тоже зачастую оснащены собственными специфическими языками программирования сверхвысокого уровня. Разумеется, это позволяет расширить область применения системы за пределы возможностей встроенных средств или комбинировать упомянутые средства для решения задач повышенной сложности, а также создавать собственные функции и команды.

Имеется целый ряд причин, нередко заставляющих пользователей программных средств отказываться от применения их «втемную» [54]:

1. *В открытую программу можно вносить изменения и дополнения, расширяющие сферу ее применения или повышающие ее эффективность для частных случаев*.

2. *В открытой программе можно исправить обнаруженную ошибку разработчика*.

3. *Тексты программ в идеале должны быть доступны обучающимся их использованию* – эта причина лежит в педагогической сфере.

В настоящее время появляются инструментальные системы и среды, при работе с которыми пользователь создает собственные библиотеки моделей и методов обработки данных. Следует отметить, что, как правило, начинающий программист зачастую недооценивает проблемы отладки и преимущества универсальности и, поэтому, он пишет собственные процедуры, тогда как опытный прежде ищет ранее созданные.

Вообще, программирование – это, прежде всего, создание логического алгоритма для достижения желаемой цели. И в этом плане в ряде случаев с некоторых позиций отчасти неважно, будет он реализован на ЭВМ или с помощью «ручных» способов. Другая сторона проблемы заключается в том, что даже если инженер будет ставить задачу профессиональному разработчику (а не писать приложения самостоятельно), то знание основ программирования поможет корректно и правильно сформулировать задание и

вообще понимать работу «смежника», говорить с ним на одном языке. Более того, довольно часто специалисту в какой-то предметной области легче освоить программирование, чем заставить разработчика программ «погрузиться» в прикладную тему (формального технического задания тут, как правило, недостаточно). В этом смысле следует особо выделить научных работников, занимающихся, например, исследованиями в области математического моделирования, где построение и изучение математических моделей довольно часто находится в неразрывной связи с процессом создания и отладки программы [33].

Все высказанные выше соображения подтверждают необходимость изучения будущими инженерами вычислительной математики и программирования, являющихся фундаментом методов обработки информации.

§ 1.2. Понятие о современных системах программирования

Итак, несмотря на многообразие компьютерных программ, пользователям может потребоваться что-то такое, чего не делают (или делают, но не так) имеющиеся программы. В таких случаях и следует использовать системы программирования, т.е. системы для разработки новых программ.

Важнейшими требованиями к программному обеспечению являются надежность (корректность и устойчивость в процессе эксплуатации), производительность разработки, дружественность к пользователю, эффективность и переносимость на другие платформы. Все эти взаимозависимые требования должны «закладываться» в программный проект уже на самых ранних его этапах, и исключительно важно именно с этих позиций сделать рациональный выбор удобной системы программирования.

Системы программирования в общем случае включают [54]:

– *редактор* (инструментальное средство для создания и изменения файлов с текстом программы в исходном коде (тексте));

– *компилятор* (выполняет преобразования исходного текста в объектный код) или *интерпретатор* (в отличие от компилятора непосредственно выполняет исходный код);

– *библиотеки полезных программ* (соответствующий *библиотекарь* поддерживает работу (добавление, удаление, обновление) с ранее полученными совокупностями объектных файлов);

– *линкер* (он же компоновщик, редактор связей – собирает вместе объектные файлы отдельных компонент программы и системные подпрограммы и разрешает внешние ссылки между ними, формируя исполняемый файл);

– *загрузчик* (копирует исполняемый файл с диска в оперативную память и инициализирует компьютер перед выполнением программы);

– *отладчик* (он же дебаггер – дает возможность управлять выполнением программы на уровне отдельных операторов исходного кода для диагностики ошибок);

– *профилировщик* (замеряет, сколько времени затрачивается на каждую компоненту программы, что необходимо для последующего совершенствования программы);

– *средства тестирования* (автоматизируют процесс тестирования программ, создавая и выполняя тесты и анализируя результаты тестирования);

– *средства конфигурирования* (автоматизируют создание программ и прослеживают изменения до уровня файлов с исходным текстом программ; обновляют объектные файлы, для которых изменились исходные модули; контролируют версии и т.д.);

– *различные вспомогательные программы*.

Разумеется, при выборе языка программирования следует также учитывать особенности и возможности соответствующей системы программирования. Заметим, что в настоящее время почти все системы программирования для персональных компьютеров представляют собой интегрированные среды с оконным интерфейсом, приблизительно одинаковыми развитыми возможностями работы с файлами, редактирования и набора текста программы. Для удобства пользователя набор текста может сопровождаться автоматическим выделением цветом, контролем правильности (и парности) ключевых слов, выдерживанием стандартных позиций, дополнительными отступами при вложенных конструкциях и т.д. В современных системах программирования содержится большая часть перечисленных выше составных частей, каждая из которых инициируется нажатием соответствующей комбинации клавиш или выбором соответствующего пункта меню.

§ 1.3. Обзор конструкций современных языков программирования

1.3.1. Некоторые предварительные замечания.

Одним из наиболее важных факторов, которые влияют на качество создаваемого программного продукта, является именно язык программирования. Его рациональная организация и четко продуманная система понятий, достаточно общих, но в то же время простых и ясных, хорошо взаимодействующих между собой, помогает пользователю упорядочить свои мысли, отделить на каждом этапе работы главное от второстепенного и последовательно продвигаться к намеченной цели. С сожалением приходится констатировать, что многие программисты страстно любят свой «родной» язык программирования и неспособны ни проанализировать конструкции языка в сравнении, ни оценить преимущества и недостатки разных современных языков и языковых понятий. Те или иные конструкции языка зачастую используются без должного отбора, практически без учета их надежности и эффективности. Ниже представлен обзор некоторых конструкций современных языков программирования.

1.3.2. Архитектура языка.

Архитектуру языка определяют его синтаксис, семантика и прагматика. Введем соответствующие определения с необходимыми пояснениями [54].

Синтаксис языка программирования – это сторона языка программирования, которая описывает структуру программ как наборов символов (обычно говорят – безотносительно к содержанию). Каждый язык программирования имеет синтаксическое описание. Обычно синтаксис языка опре-

деляют посредством правил Бэкуса-Наура (формальная система описания синтаксиса, в которой одни синтаксические категории последовательно определяются через другие категории).

Семантика языка программирования, вообще говоря, противопоставляется синтаксису. Синтаксис языка описывает «чистый» язык, в то же время семантика приписывает значения (действия) различным синтаксическим конструкциям (иными словами, это правила истолкования). Имеется несколько подходов к определению семантики языков программирования, причем наиболее широко распространены разновидности следующих трех: операционного, денотационного (математического) и деривационного (аксиоматического). При описании семантики в рамках операционного подхода обычно исполнение конструкций языка программирования интерпретируется с помощью некоторой воображаемой (абстрактной) ЭВМ. Деривационная семантика описывает последствия выполнения конструкций языка с помощью языка логики и задания предусловий и постусловий. Денотационная семантика оперирует понятиями, типичными для математики (множества, соответствия, а также суждения, утверждения и др.).

Прагматика языка программирования – это сопоставление текстов целям и намерениям автора или, по сути дела, методология программирования (описание принципов, методов и приемов, позволяющих, исходя из постановки задачи, составить программу ее решения).

Архитектуру языка оценивают по его концептуальной продуманности, имея в виду цельность (возможность предсказать одни решения авторов языка по другим), модульность, ортогональность и т.п. Детали проекта в идеале должны быть следствием относительно небольшого числа базисных ключевых решений. Иными словами, в язык программирования следует включать не все, что может потребоваться, а только абсолютно необходимое.

1.3.3. Типы данных.

Тип данных определяет множество значений, которые могут принимать экземпляры принадлежащих к нему данных, представление этих значений, правила доступа к ним и операции над ними. Введение типа данных в язык программирования позволяет получить более короткую программу, которую легче создавать, понимать, проверять и изменять. Использование типов данных обеспечивает надежность программы (в плане защиты от ошибок, связанных с некорректным присваиванием, некорректными операциями, некорректной передачей параметров, несоответствием типов и т.д.) и повышает степень стандартизации программного продукта (благодаря соглашениям о типах, поддерживаемых большинством систем программирования, программисты могут быстро менять свои рабочие инструменты, а программы не требуют при этом больших переделок при переносе исходных текстов в другую среду).

Типы данных различаются, начиная с нижних уровней системы. Концепция типа данных появилась в языках программирования высокого уровня как естественное отражение того факта, что обрабатываемые программой данные могут иметь различные множества допустимых значений, хра-

ниться в памяти компьютера различным образом, занимать различные объемы памяти и обрабатываться с помощью различных команд процессора.

Каждый язык программирования поддерживает один или несколько встроенных типов данных (базовых типов). Развитые языки программирования помимо этого дают программисту возможность вводить и описывать собственные типы данных (производные), комбинируя или расширяя существующие. Классическими встроенными типами являются числовые типы (целые и вещественные) разной длины и соответственно диапазона значений. Очевидно, что для многих приложений крайне желательно иметь встроенные типы комплексных переменных, чем, однако, могут похвастать лишь очень немногие языки программирования (например, Фортран).

Вообще, следует пояснить, что типы данных в языках программирования далеко не всегда строго соответствуют подобным математическим типам. Так, например, тип «целое число» большинства языков программирования не соответствует принятому в математике типу «целое число», так как в математике указанный тип не имеет ограничений ни сверху, ни снизу, а в языках программирования такие ограничения имеются. Как правило, в языках и системах имеется множество целых типов, отличающихся допустимым диапазоном значений (определяемым объемом занимаемой памяти).

1.3.4. О представлении числовых типов.

Во многих языках программирования средства явного управления диапазоном и точностью числовых данных отсутствуют. Как правило, можно лишь указать, например, что требуется «двойная точность» (в некоторых случаях градаций больше), но сама эта точность зачастую зависит от реализации. Для некоторых приложений подобный стандартный набор вариантов оказывается недостаточным, что ограничивает переносимость программ. По этой причине, а также благодаря появлению относительно эффективной реализации вычислений с произвольной заданной точностью в ряде новых языков появилась возможность определения соответствующих спецификаций представления [54].

1.3.5. Структурные конструкции.

Мощным средством борьбы со сложностью программы является ее структуризация, причем это относится как к самим данным, так и к программам их обработки [54].

Структурированными данными являются массивы, множества, записи, таблицы, списки, очереди. Их можно классифицировать на статические (соответствующее распределение памяти производится в процессе компиляции; время существования таких данных совпадает с полным временем выполнения всей программы), квазистатические (время существования согласуется с временем очередного исполнения их статически определимой области действия) и динамические (распределение памяти производится в процессе счета). Во многих приложениях требуются именно динамические структуры данных, однако стоит отметить, что применение последних сопряжено с более сложной диагностикой некоторых часто встречающихся ошибок (например, неконтролируемый выход индекса за пределы диапазона).

Если язык программирования не предоставляет соответствующих средств, динамические объекты, как правило, моделируются с помощью квазистатических (или их частей) или статических. Истинно динамические объекты обладают двумя особенностями. Во-первых, они создаются при выполнении так называемых генераторов, а не при обработке объявлений, а во-вторых, доступ к ним осуществляется через объекты ссылочных типов.

Структуризация программы достигается свободным форматом ее записи (с выявлением вложенности конструкций) и применением структурированных операторов.

1.3.6. Уровень языка.

Языки программирования, по сути, служат своеобразным мостом между аппаратными средствами и реальным миром. Есть неизбежное противоречие между высокими уровнями абстракции, которые легче понять и безопаснее использовать, и низшими уровнями, более гибкими и зачастую допускающими более эффективную реализацию. В этой связи какой-то язык может подходить для одного проекта и не подходить для другого.

В современной практике программирование в машинных кодах ведется все реже. Такая практика, как правило, считается оправданной лишь при написании ядер операционных систем, драйверов устройств или компиляторов для нового типа машин. Программирование на языках более высокого уровня дает ряд существенных преимуществ [54]:

1. Программист избавлен от учета особенностей компьютера, сами собой устраняются многочисленные ошибки в кодах команд, назначении регистров, выравнивании операндов на границу и т.д.

2. Программа мобильна, т.е. применима на разных типах компьютеров и в различных операционных системах.

3. Имеется возможность укрупненно описывать операции над сложными структурами (например, массивами).

4. Программы становятся хорошо обозримыми, «читаемыми», в них легче обнаружить алгоритмические ошибки.

5. Вследствие большей семантической емкости конструкций языка повышается производительность программирования.

6. Облегчается понимание программы, ее применение и сопровождение.

Заметим, что характерным примером или, быть может, даже важнейшим показателем информационной емкости для вычислительных приложений является наличие в них операций над матрицами в целом (это характерно, в частности, для современных реализаций языка Фортран).

1.3.7. Процедуры.

Процедура – это поименованная или иным образом идентифицированная часть программы (самостоятельная программная единица). Процедура может быть многократно вызвана из различных частей программы или других процедур. В языках программирования для оформления и использования процедур существуют специальные синтаксические средства.

Процедуры изначально появились как средство оптимизации программ по объему занимаемой памяти – они позволили не повторять в программе

идентичные блоки кода, а описывать их однократно и вызывать по мере необходимости. В настоящее время данная функция процедур отчасти стала вспомогательной, главное их назначение – структуризация программы с целью удобства ее понимания и сопровождения. Выделение набора действий в процедуру и вызов ее по мере необходимости позволяет логически выделить целостную подзадачу, имеющую типовое решение. Помимо экономии памяти такой прием дает еще одно преимущество перед повторением однотипных действий: любое изменение (исправление ошибки, оптимизация, расширение функциональности), сделанное в процедуре, автоматически отражается на всех ее вызовах, в то время как при дублировании каждое изменение необходимо вносить в каждое вхождение изменяемого кода. Это оправдано даже в тех случаях, когда в процедуру выделяется однократно производимый набор действий (можно сократить размеры целостных блоков кода, составляющих программу, т.е. сделать программу более понятной и обозримой).

Процедура вызывается с передачей ей набора *параметров* (фактических). Параметры задаются, чтобы задавать различные варианты ее выполнения, передать ей исходные данные (входные данные программы) и получить результаты вычисления (выходные данные). Существует несколько способов передачи параметров в процедуру: по значению, по ссылке, по имени, через стек. Механизмы передачи параметров в разных языках (и для разных видов параметров) сильно различаются, и от их понимания программистами во многом зависят надежность и эффективность программ.

Залог надежного программирования состоит в соответствии (совпадении) типов *формальных* и *фактических* (фактически задаваемых) параметров. Существуют различные варианты контроля такого соответствия.

Весьма ценной, но достаточно редкой является работа с необязательными и ключевыми параметрами. Первая возможность позволяет задавать значения аргументов «по умолчанию», а вторая исключает ошибки, связанные с перестановками аргументов. Некоторые языки программирования допускают описание вложенных процедур, т.е. размещение процедур внутри других процедур. Такие вложенные процедуры могут использоваться только в той процедуре, в которой они описаны. Никаких принципиальных преимуществ такое вложение процедур не дает, но может быть удобно для более логичной структуризации программы.

1.3.8. Модульность.

Модульность, являющаяся важным арсеналом программирования, повышающим надежность больших программных комплексов, – это принцип, согласно которому программа разделяется на отдельные именованные объекты, называемые модулями. Модульность часто является средством упрощения задачи проектирования программы и более рациональной организации работ по ее созданию (распределение между группами разработчиков, организация централизованного контроля, тестирования и др.). При разбиении программы на модули для каждого модуля определяются реализуемые им функции, а также связи с другими модулями. Модуль, по сути, задает

Оглавление