



**А.Б. Золотов, П.А. Акимов, В.Н. Сигоров, М.Л. Мозгаева**

# **ИНФОРМАТИКА**

**программирование**

**численные методы**

**приложения в строительстве**



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

# **ИНФОРМАТИКА**

**ВТОРОЕ ИЗДАНИЕ,  
ПЕРЕРАБОТАННОЕ И ДОПОЛНЕННОЕ**

*Рекомендовано Учебно-методическим объединением  
высших учебных заведений Российской Федерации  
по образованию в области строительства  
в качестве учебника для студентов, обучающихся  
по направлению подготовки «Строительство» (бакалавриат)  
и специальности «Строительство уникальных зданий  
и сооружений» (специалитет)*



Издательство АСВ  
Москва  
2013

**Рецензенты:**

заведующий кафедрой «Строительная механика и вычислительные технологии»  
Пермского национального исследовательского политехнического университета  
советник РААСН, доктор технических наук, профессор *Г.Г. Каишеварова*

профессор кафедры «Техническая механика»

Ростовского государственного строительного университета

советник РААСН, доктор технических наук, профессор *П.П. Гайджуров*

руководитель Научно-образовательного центра компьютерного моделирования  
уникальных зданий, сооружений и комплексов

Московского государственного строительного университета

член-корреспондент РААСН, доктор технических наук, профессор *А.М. Белостоцкий*

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

Информатика.: Учебник. 2-е изд., перераб. и доп.. – М.: Издательство АСВ, 2013.  
– 400 с.

ISBN 978-5-93093-982-8

Содержание книги подразделено на три главы и четыре приложения, в которых приводятся сведения о современных языках, системах программирования и прикладном программном обеспечении, рассматриваются основы численных методов и описываются их приложения к решению конкретных прикладных задач, главным образом строительной направленности.

Учебник предназначен для студентов, обучающихся по направлению подготовки «Строительство» (бакалавриат) и специальности «Строительство уникальных зданий и сооружений» (специалитет), 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 978-5-93093-982-8

© Издательство АСВ, 2013

© А.Б. Золотов, П.А. Акимов,

В.Н. Сидоров, М.Л. Мозгалева, 2013

---

Учебник

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

**ИНФОРМАТИКА**

Компьютерный набор и верстка: П.А. Акимов, М.Л. Мозгалева

Редактор: В.Ш. ММерзлякова

Дизайн обложки: Н.С. Романова

Лицензия ЛР №0716188 от 01.04.98. Подписано к печати 07.08.13 Формат 60х90/16

Бумага офс. Гарнитура Таймс. Печать офсетная.

Усл. 25,0 п.л. Тираж 500 экз. Заказ № \_\_\_\_\_

ООО «Издательство АСВ», 129337, г. Москва, Ярославское шоссе, д. 26,  
отдел реализации – оф. 511 тел./факс: +7(499) 183-56-83, e-mail: [iasv@mgsu.ru](mailto:iasv@mgsu.ru), <http://www.iasv.ru/>

## Предисловие ко второму изданию

---

Первое издание этой книги вышло в свет в издательстве АСВ под общей редакцией А.Б. Золотова в 2010 году [64]. За это время стало ясно, что она нашла своего читателя. Книга достаточно активно используется студентами, аспирантами, преподавателями не только в Московском государственном строительном университете (МГСУ), но и в большом количестве других вузов страны. В тоже время обнаружился ряд опечаток и неточностей в изложении, и появились свежие идеи по улучшению подачи материала. Поэтому авторы с благодарностью приняли любезное предложение издательства АСВ о переиздании учебника.

Во втором издании исправлены замеченные недостатки и добавлена новая глава, посвященная вопросам решения прикладных задач в системе MATLAB, произведена некоторая перестановка материала для облегчения чтения.

Любым учебником обычно пользуются как студент, так и преподаватель – каждый для своих целей. Сначала и тот, и другой заинтересованы иметь книгу, где, помимо формально необходимого минимума теории, имеются по возможности разнообразные содержательные примеры ее использования, пояснения, научные комментарии, демонстрируются взаимосвязи, указываются перспективы развития. Но в момент подготовки к экзамену или зачету студент желает видеть тот материал, который выносится на экзамен или зачет. Преподаватель точно также, завершая подготовку курса, отбирает только тот материал, который может и должен быть изложен в отведенное курсу время. В этой связи следует иметь в виду, что текст данного учебника, конечно, заметно шире того конспекта лекций, на базе которого он написан.

К сожалению, в 2008 году скоропостижно ушел из жизни один из авторов книги – Александр Борисович Золотов, выдающийся ученый, математик, механик и педагог, доктор технических наук, профессор, Почетный член Российской академии архитектуры и строительных наук, Почетный работник высшего образования России, Почетный профессор МГСУ. Однако подготовка второго издания к печати была выполнена в том числе с учетом его замечаний и рекомендаций.

Когда книга уже написана, видишь как много в ней недостатков, как много мест следовало бы в ней переделать. Однако приходится смириться и отложить переделки до возможного переиздания. В связи с этим просим читателей направлять нам критические замечания и пожелания, к которым мы отнесемся со всем необходимым вниманием.

Авторы верят, что новая версия учебника окажется столь же полезной для читателей, как и предыдущая.

Прошло относительно немного времени с тех пор, как в школах и вузах страны начали изучать дисциплину «Информатика». В настоящее время эта дисциплина 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 информа-

тику как сугубо практическую дисциплину, учащую не знаниям, а умению «правильно нажимать кнопки» персонального компьютера, работая в наиболее распространенных программах, и осваивая лишь их офисные приложения. Надеемся, что читатель с иронией отнесется и к такой трактовке. Достижение профессиональных компетенций при столь упрощенном 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 вычислительных 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 образовательного процесса по дисциплине «Информатика» в техническом вузе:

- 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 актуальных профессиональных инженерных задач, основанную «на развитии в широком применении методов математического моделирования и вычислительного эксперимента и служащую ближайшим стратегическим резервом ускорения научно-технического прогресса» [137].

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, прежде всего, с тех позиций, что все вышеперечисленное является основой школьных курсов дисциплины «Информатика».

Учебник разбит на три главы, отражающие следующие аспекты информатики: обзор современных языков и систем программирования, основы программирования, основы численных методов решения задач линейной алгебры, численные методы решения инженерных задач и их реализации на языках и системах программирования.

Принятый авторами стиль изложения во многом определяется характером будущей работы студентов. Для них информатика, численное и компь-

ютерное 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 вузов.

Весь теоретический материал снабжен большим количеством примеров, которые носят как поясняющий, так и контрольный характер. Заметим, что задания на выполнение практических и лабораторных работ, представленных в Главе 3, 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.

Авторы благодарны рецензентам книги, профессорам Г.Г. Кашеваровой (Пермский национальный исследовательский политехнический университет), П.П. Гайджуру (Ростовский государственный строительный университет) и А.М. Белостоцкому (МГСУ), за проявленный интерес и доброе отношение к работе.

Для оформления обложки книги использовались иллюстрации, любезно предоставленные профессором А.М. Белостоцким.

Большую помощь в подготовке книги авторам оказала генеральный директор издательства Ассоциации строительных высших учебных заведений (Издательство АСВ), профессор Н.С. Никитина. Также благодарим сотрудников Издательства АСВ за содействие в подготовке книги к изданию. В подготовке второго издания книги также принимали участие В.Н. Варапаев, С.Б. Пеньковой, В.А. Харитонов, и особенно большой вклад принадлежит Т.Б. Кайтукову. К сожалению, мы не имели возможности в полной мере использовать все их пожелания, постараемся это сделать впоследствии.

Издание настоящего учебного пособия осуществлено в рамках выполнения следующих международных образовательных проектов:

1. Проект 530793-TEMPUS-1-2012-1-SE-TEMPUS-JPCR «Магистерские программы по возобновляемой энергии и энергетической эффективности зданий в Центральной Азии и России».
2. Проект 530603-TEMPUS-1-2012-1-LT-TEMPUS-JPCR (2012-3071/001-001) «Реформирование программ в сфере градостроительства на пространстве Восточного соседства».

*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

В последние десятилетия достижения человека во всех сферах его деятельности, в том числе в области науки и техники, обеспечиваются эффективными способами поиска, сбора, обмена, хранения и переработки 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, контроль состояния (мониторинг), 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 [107,156].

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» [141]. 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 раз-

работку 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 формируемой 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, воз-



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, 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, используемые для записи алгоритмов и обеспечивающие им наличие всех необходимых свойств, называются *алгоритмическими языками*.

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

## ЯЗЫКИ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Данная глава посвящена важнейшему разделу информатики – программированию на ЭВМ. Наиболее подробно ниже рассматриваются язык программирования FORTRAN и система программирования MATLAB, являющиеся в настоящее время самым эффективным средством для решения инженерных задач и отражающие различные тенденции в современном программировании. Приводятся все необходимые сведения и примеры, призванные сформировать у читателя общую картину, сделав возможным самостоятельное решение относительно несложных задач по программированию. При необходимости изучить тот или иной язык или систему программирования «полностью» (задача эта весьма неопределенная хотя бы из-за наличия у каждого языка или среды программирования многих версий, особенностей трансляторов, интерфейсных оболочек и т.д.) необходимо обратиться к специализированной литературе.

### § 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 [6,52,53,80,93,122]) 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» [132]:

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 некоторыми позициями 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.

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, дружелюбность 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 [132]:

– *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 последова-

тельно 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 [132].

*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 определяют 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, храниться 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 [132].

### 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 [132].

*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 определенной 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 [132]:

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 обзримыми, «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 обзримой).

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.



## Оглавление

	стр.
<b>Предисловие ко второму изданию.</b> .....	<b>3</b>
<b>Основные понятия информатики.</b> .....	<b>6</b>
<b>Введение.</b> .....	<b>9</b>
<b>Глава 1. Современные языки и системы программирования.</b> .....	<b>12</b>
§ 1.1. О роли и необходимости изучения языков программирования. ....	12
§ 1.2. Понятие о современных системах программирования. ....	13
§ 1.3. Обзор конструкций современных языков программирования. ....	14
1.3.1. Некоторые предварительные замечания. ....	14
1.3.2. Архитектура языка. ....	15
1.3.3. Типы данных. ....	16
1.3.4. О представлении числовых типов. ....	16
1.3.5. Структурные конструкции. ....	17
1.3.6. Уровень языка. ....	17
1.3.7. Процедуры. ....	18
1.3.8. Модульность. ....	19
1.3.9. Стандартизация. ....	20
§ 1.4. Понятие об объектно-ориентированном программировании. ....	20
§ 1.5. Свойства языков программирования и требования к ним. ....	21
§ 1.6. Технологии производства программ. ....	22
§ 1.7. Основы программирования на языке Фортран (FORTRAN). ....	23
1.7.1. Краткая характеристика и история развития языка программирования FORTRAN. ....	23
1.7.2. Алфавит языка Фортран. ....	38
1.7.3. Структура главной программы. Форматы записи. ....	39
1.7.4. Имена. ....	41
1.7.5. Объекты данных. ....	41
1.7.6. Операции и выражения. ....	49
1.7.7. Встроенные математические функции. ....	54
1.7.8. Метки и комментарии. ....	57
1.7.9. Оператор присваивания. ....	58
1.7.10. Простой ввод-вывод. ....	59
1.7.11. Оператор и конструкции IF. Конструкция SELECT CASE. ....	62
1.7.12. Операторы STOP, PAUSE, GOTO, CONTINUE. ....	69
1.7.13. Производные типы данных. ....	70
1.7.14. Операторные функции. ....	73
1.7.15. Циклы. ....	74
1.7.16. Форматный ввод-вывод данных. ....	80
1.7.17. Массивы. ....	90
1.7.18. Программные компоненты. ....	120
§ 1.8. Основы программирования в системе MATLAB. ....	153
1.8.1. Краткая характеристика и история развития системы MATLAB. ....	153
1.8.2. Вычисление арифметических выражений. ....	155

1.8.3.	Векторно-матричные операции. ....	158
1.8.4.	Построение графиков. ....	163
1.8.5.	Логические операции. ....	164
1.8.6.	Условный оператор IF. ....	166
1.8.7.	Операторы цикла. ....	167
1.8.8.	Типы М-файлов. ....	167
<i>Лабораторный практикум к Главе 1</i>		
Лабораторная работа 1.1.	Запись арифметических выражений. ....	169
Лабораторная работа 1.2.	Вычисление корней квадратного уравнения. ...	174
Лабораторная работа 1.3.	Запись логических выражений. ....	175
Лабораторная работа 1.4.	Ветвящиеся алгоритмы. ....	177
Лабораторная работа 1.5.	Определение наибольшего и наименьшего значения функции на отрезке и построение ее графика. ....	179
Лабораторная работа 1.6.	Вычисление суммы. ....	182
Лабораторная работа 1.7.	Одномерные массивы. ....	185
Лабораторная работа 1.8.	Вычисление скалярного произведения векторов. ....	189
<b>Глава 2. Основы численных методов. ....</b>		<b>194</b>
§ 2.1.	Основные понятия линейной алгебры. ....	194
2.1.1.	Линейное пространство. Евклидово пространство. ....	194
2.1.2.	Арифметическое пространство. Векторы и операции над ними. Линейная зависимость векторов. Базис. ....	195
2.1.3.	Матрицы и операции над ними. Некоторые виды матриц. ....	197
2.1.4.	Скалярное произведение векторов. ....	201
2.1.5.	Норма вектора. Некоторые свойства векторных норм. Абсолютная и относительная погрешности. Сходимость. ....	202
2.1.6.	Норма матрицы. Некоторые свойства матричных норм. ...	203
2.1.7.	Число обусловленности матрицы. ....	205
§ 2.2.	Прямые методы решения систем линейных алгебраических уравнений. ....	205
2.2.1.	Общий вид системы линейных алгебраических уравнений (СЛАУ). Понятие о прямых методах решения СЛАУ. ....	205
2.2.2.	Метод Гаусса. ....	207
2.2.3.	Вычисление определителя матрицы коэффициентов системы. Некоторые свойства определителей. ....	209
2.2.4.	Выбор главного элемента. ....	209
2.2.5.	LU-разложение. ....	210
2.2.6.	Итерационное уточнение. ....	212
2.2.7.	Метод Гаусса для ленточной СЛАУ. ....	212
2.2.8.	СЛАУ с несколькими правыми частями. ....	213
2.2.9.	Вычисление обратной матрицы. ....	213
§ 2.3.	Итерационные методы решения систем линейных алгебраических уравнений. ....	214
2.3.1.	Понятие об итерационных методах решения СЛАУ. ....	214
2.3.2.	Метод Зейделя и метод простой итерации. ....	215
2.3.3.	Общий вид итерационного процесса. ....	216
§ 2.4.	Методы вычисления собственных значений и собственных векторов матриц. ....	219

2.4.1.	Основные определения. ....	219
2.4.2.	Характеристическое уравнение. ....	220
2.4.3.	Понятие о методах решения проблемы собственных значений. ....	222
2.4.4.	Определение максимального по модулю собственного значения степенным методом. ....	223
2.4.5.	Собственные значения обратной матрицы. ....	225
2.4.6.	Обобщенная проблема собственных значений. ....	225
2.4.7.	Вычисление минимального собственного числа степенным методом. ....	226
§ 2.5.	Методы численного интегрирования. ....	226
2.5.1.	Понятие о формулах численного интегрирования. ....	226
2.5.2.	Формула прямоугольников (формула средних). ....	227
2.5.3.	Формула трапеций. ....	228
2.5.4.	Формула Симпсона. ....	229
2.5.5.	Точность квадратурных формул. ....	231
2.5.6.	Вычисление интеграла с заданной точностью. ....	232
2.5.7.	О современных методах численного интегрирования. ....	232
§ 2.6.	Методы решения нелинейных уравнений. ....	233
2.6.1.	Уравнения с одним неизвестным (скалярные). ....	233
2.6.2.	Решение систем нелинейных уравнений. ....	237
§ 2.7.	Метод наименьших квадратов. ....	240
2.7.1.	Понятие о методе наименьших квадратов (МНК). ....	240
2.7.2.	Применение МНК для решения переопределенных систем линейных алгебраических уравнений. ....	243
<i>Лабораторный практикум к Главе 3</i>		
<b>Лабораторная работа 2.1.</b>	Решение системы линейных алгебраических уравнений методом Гаусса. ....	245
<b>Лабораторная работа 2.2.</b>	Вычисление обратной матрицы и определителя методом Гаусса. ....	246
<b>Лабораторная работа 2.3.</b>	Решение систем линейных алгебраических уравнений итерационными методами. ....	249
<b>Лабораторная работа 2.4.</b>	Вычисление собственных значений и собственных векторов симметричной матрицы. ....	252
<b>Лабораторная работа 2.5.</b>	Численное интегрирование. ....	255
<b>Лабораторная работа 2.6.</b>	Вычисление корня нелинейного уравнения. ...	260
<b>Лабораторная работа 2.7.</b>	Построение оптимальной прямой по методу наименьших квадратов. ....	262
<b>Глава 3. Численные и численно-аналитические методы решения прикладных задач. ....</b>		<b>264</b>
§ 3.1.	Численное решение краевых задач. ....	264
3.1.1.	Некоторые предварительные понятия и примеры. ....	264
3.1.2.	Общий вид краевой задачи на отрезке для дифференциального уравнения второго порядка. ....	265
3.1.3.	Метод конечных разностей. ....	266
3.1.4.	Порядок аппроксимации производных. ....	268
§ 3.2.	Численное решение задачи об устойчивости сжатого стержня. ....	269
3.2.1.	Некоторые предварительные понятия. ....	269

	3.2.2.	Решение методом конечных разностей. ....	270
	3.2.3.	Вычисление минимальной критической силы степенным методом. ....	271
§ 3.3.		Численное решение краевой задачи для уравнения Пуассона. ....	272
	3.3.1.	Некоторые предварительные понятия и пример. ....	272
	3.3.2.	Общий вид формулировки краевой задачи для уравнения Пуассона. ....	273
	3.3.3.	Решение задачи Дирихле методом конечных разностей. ...	274
§ 3.4.		Численное решение задачи Коши (задачи с начальными условиями). ....	276
	3.4.1.	Некоторые предварительные понятия и примеры. ....	276
	3.4.2.	Метод Эйлера. ....	278
	3.4.3.	Понятие об устойчивости разностной схемы. ....	280
§ 3.5.		Численное решение задачи теплопроводности. ....	281
	3.5.1.	Некоторые предварительные понятия и пример. ....	281
	3.5.2.	Математическая формулировка задачи. ....	282
	3.5.3.	Решение задачи методом конечных разностей. ....	283
§ 3.6.		Решение задачи линейного программирования. ....	286
	3.6.1.	Понятие о задаче математического программирования. ...	286
	3.6.2.	Формулировка задачи линейного программирования. ....	286
	3.6.3.	Пример формулировки и решения задачи линейного программирования. ....	287
	3.6.4.	Некоторые широко употребляемые термины задач линейного программирования. ....	289
	3.6.5.	Некоторые важные замечания. ....	290
§ 3.7.		Метод конечных элементов (МКЭ) (на примере краевой задачи для обыкновенного дифференциального уравнения изгиба растянуто-изогнутой балки). ....	290
	3.7.1.	Соответствие задач определения минимума функционала и решения системы уравнений. ....	290
	3.7.2.	Вариационная постановка задачи об изгибе растянуто-изогнутой балки. ....	291
	3.7.3.	Решение вариационной задачи. ....	292
§ 3.8.		Вычисление функций от матриц. ....	295
	3.8.1.	Определения функции от матрицы. ....	295
	3.8.2.	Некоторые важные примеры приложений. ....	299
§ 3.9.		Универсальный метод построения матриц жесткости и векторов нагрузок конечных элементов при решении одномерных задач расчета конструкций. ....	299
	3.9.1.	Введение. ....	299
	3.9.2.	Некоторые предварительные построения. ....	300
	3.9.3.	Аппроксимация основных неизвестных на конечном элементе. ....	304
	3.9.4.	Универсальный алгоритм построения матриц жесткости конечного элемента. ....	308
	3.9.5.	Универсальный алгоритм построения вектора нагрузок конечного элемента. ....	312
§ 3.10.		Численно-аналитическое решение задачи теплопроводности. ....	313
§ 3.11.		Численно-аналитическое решение задачи о колебаниях балки при ударе. ....	313

	3.11.1. Математическая формулировка задачи. ....	313
	3.11.2. Дискретно-аналитический метод решения задачи. ....	315
§ 3.12.	Аналитический метод вычисления геометрических характеристик поперечных сечений элементов конструкций, использующий теорию обобщенных функций. ....	318
	3.12.1. Предварительные сведения. ....	318
	3.12.2. Геометрические характеристики сечений. ....	320
	3.12.3. Вычисление геометрических характеристик для сечения в виде произвольного многоугольника. ....	321
	3.12.4. Вычисление геометрических характеристик для произвольного сечения. ....	324
	<i>Лабораторный практикум к Главе 4</i>	
Лабораторная работа 3.1.	Решение краевой задачи методом конечных разностей. ....	325
Лабораторная работа 3.2.	Решение задачи об устойчивости сжатого стержня. ....	328
Лабораторная работа 3.3.	Решение задачи Дирихле для уравнения Пуассона. ....	332
Лабораторная работа 3.4.	Решение задачи об изгибе консоли (задача Коши). ....	336
Лабораторная работа 3.5.	Решение задачи теплопроводности. ....	337
Лабораторная работа 3.6.	Решение задачи линейного программирования. ....	342
Лабораторная работа 3.7.	Решение задачи об изгибе растянуто-изогнутой балки методом конечных элементов. ....	344
Лабораторная работа 3.8.	Вычисление функций от матрицы. ....	347
Лабораторная работа 3.9.	Решение задачи о поперечном изгибе балки на упругом основании методом конечных элементов. ....	351
Лабораторная работа 3.10.	Численно-аналитическое решение задачи теплопроводности. ....	356
Лабораторная работа 3.11.	Решение задачи о колебаниях балки при ударе. ....	356
Лабораторная работа 3.12.	Вычисление геометрических характеристик сечения. ....	361
Приложение 1.	<b>Основы работы с Intel Visual Fortran Compiler. ....</b>	<b>365</b>
	П.1.1. Предварительные сведения. ....	365
	П.1.2. Создание проекта. ....	365
	П.1.3. Создание файла с исходным текстом программы. ....	365
	П.1.4. Добавление файла в проект и удаление. ....	367
	П.1.5. Запуск программы. ....	367
Приложение 2.	<b>Некоторые математические библиотеки языка программирования Фортран. ....</b>	<b>370</b>
	<i>Часть 1. Математическая библиотека SSP.</i>	
	П.2.1. Общая характеристика математической библиотеки SSP. ....	370

П.2.2.	Подпрограмма SIMQ – решение системы линейных алгебраических уравнений. ....	372
П.2.3.	Подпрограмма MINV – нахождение обратной матрицы. ....	373
П.2.4.	Подпрограмма NROOT – решение обобщенной проблемы собственных значений. ....	373
П.2.5.	Подпрограмма QATR – вычисление определенного интеграла. ....	374
П.2.6.	Подпрограмма SIMPLPR – решение задачи линейного программирования. ....	374
<b>Часть 2. Математическая библиотека IMSL.</b>		
П.2.7.	Общая характеристика математической библиотеки IMSL. ....	375
П.2.8.	Подпрограмма LSLRG – решение системы линейных алгебраических уравнений. ....	378
П.2.9.	Подпрограмма LINRG – нахождение обратной матрицы. ....	379
П.2.10.	Подпрограмма LFTRG – нахождение LU-разложения матрицы. ....	379
П.2.11.	Подпрограмма LFDRG – вычисление определителя матрицы. ....	380
П.2.12.	Подпрограммы EVCRG, EVCSF – решение проблемы собственных значений. ....	380
П.2.13.	Подпрограммы GVCRG, GVCSF – решение обобщенной проблемы собственных значений. ....	381
П.2.14.	Подпрограмма QDAG – численное интегрирование. ....	382
П.2.15.	Подпрограмма ZBREN – решение нелинейных уравнений. ....	383
П.2.16.	Подпрограмма DLPRS – решение задачи линейного программирования. ....	384
П.2.17.	Операторы для решения задач линейной алгебры. ....	385
<b>Приложение 3.</b>	<b>Основы работы с графическим пакетом Grapher. ....</b>	<b>386</b>
<b>Приложение 4.</b>	<b>О погрешности результата решения задачи. ....</b>	<b>386</b>
<b>Библиографический список.</b>	<b>.....</b>	<b>390</b>