Министерство образования Республики Беларусь
Учреждение образования
«Витебский государственный технологический университет»

Е.Ю. Вардомацкая, Т.Н. Окишева

Информатика в прикладных задачах легкой промышленности

Витебск

2007

УДК 004.9:67/68
ББК 32.937 - 018
В 18

Авторы: Вардомацкая Е.Ю. (1-4, 5.1-5.4, 6.1-6.3, 7.1-7.2, 8.1-8.4, 9.1-9.6, 10.1-10.5, 11.1-11.3, 12.1-12.2, 13),
Окишева Т.Н. (5.5, 6.4, 7.3, 8.4, 9.7, 10.6, 11.4, 12.3)

Рецензенты:
кафедра автоматизации научных исследований УО «ВГУ им. П. М. Машерова», заведующий кафедрой доктор технических наук, профессор Ключников А.С.;
кандидат технических наук, профессор кафедры информатики УО «ВГТУ» Шарстнев В.Л.

Рекомендовано в качестве пособия редакционно-издательским советом учреждения образования «Витебский государственный технологический университет», протокол № 7 от 20 декабря 2006 г.

Вардомацкая, Е.Ю.
В 18 Информатика в прикладных задачах легкой промышленности: учебное пособие / Е.Ю. Вардомацкая, Т.Н. Окишева. – Витебск: УО «ВГТУ», 2007. – 188 стр.

ISBN 985-481-073-9

Изложены основные вопросы технического и программного обеспечения персональных компьютеров. Рассмотрена технология выполнения прикладных экономических, инженерно-экономических и инженерно-технологических расчетов средствами языка программирования Паскаль. По каждой теме приведены вопросы и варианты задач различного уровня сложности для самостоятельной работы и закрепления полученных знаний.

Пособие предназначено для студентов технологических и экономических специальностей, а также инженерно-экономических работников текстильной и легкой промышленности.

УДК 004.9:67/68
ББК 32.937 - 018

ISBN 985-481-073-9

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Информатика, как наука, занята разработкой методологии создания информационного обеспечения процессов управления объектами на базе компьютерных информационных систем.

В современном высшем образовании на нее возлагаются задачи систематизации приемов создания, хранения, воспроизведения, обработки и передачи данных средствами вычислительной техники, а также принципов функционирования этих средств и методов управления ими.

Одной из важнейших задач при подготовке специалистов для легкой промышленности является формирование у них информационной культуры. С этой целью в учебные планы экономических и технологических специальностей вузов, которые готовят специалистов для легкой промышленности, введены такие курсы как «Информатика», «Информатика, численные методы и компьютерная графика», «Основы информатики и вычислительной техники», «Современные информационные технологии».

Цель данных курсов – научить студентов самостоятельно работать с операционной системой, грамотно ставить задачи обработки инженерно-технической и экономической информации, обоснованно выбирать технические средства для их решений, разрабатывать алгоритмы и программы на языке программирования Pascal, эффективно использовать пакеты прикладных программ для решения различных инженерно-технических и экономических задач и сформировать представление о состоянии и перспективах применения и возможностях персональных компьютеров.

Цель учебного издания – помочь студентам закрепить на конкретном материале содержание вышеназванных курсов и привить им навыки решения задач, связанных с будущей специальностью, средствами современных компьютерных технологий.

В настоящем пособии в соответствии с ныне действующими учебными программами последовательно рассматриваются данные вопросы. Здесь раскрываются базовые понятия курса: технические и программные средства персональных компьютеров, основы алгоритмизации, основы программирования на алгоритмическом языке, компьютерное моделирование инженерно-технологических задач.

Пособие состоит из тринадцати глав, объединенных в разделы.

В теме 1 «Основные сведения о персональном компьютере» (главы 2-3) приводятся основные сведения о техническом и программном обеспечении персонального компьютера.

В теме 2 «Основы алгоритмизации инженерно-технологических задач» (глава 4) излагаются основы алгоритмизации и принципы использования основных приемов алгоритмизации.

Тема 3 «Основы программирования на языке Pascal» (главы 5-10) посвящена изучению основных конструкций языка программирования Pascal и воз-

можности их использования при решении инженерно-технических, экономических или финансовых задач.

Тема 4 «Компьютерное моделирование инженерно-технологических задач» (главы 11-13) раскрывает вопросы использования элементов математического анализа, реализуемых средствами языка программирования Pascal и системы компьютерной математики Maple для решения профессиональных задач.

Каждая глава включает в себя теоретический материал, содержащий определения и разъяснения основных понятий, и вопросы для закрепления полученных знаний по теме или задания для самостоятельной работы.

При изложении материала основное внимание уделяется практике применения предлагаемых методик расчетов в различных отраслях легкой промышленности, экономико-организационным постановкам и моделированию задач, подготовке исходных данных. Предлагаемые примеры и задания разработаны как авторами пособия, так и основаны на практических материалах, представленных в целом ряде специальных изданий, предназначенных для работников данной отрасли.

Разобранные примеры и предложенные варианты заданий могут использоваться при проведении лабораторного практикума. Пособие также может быть полезным студентам других специальностей, в том числе и в процессе подготовки курсовых и дипломных работ.

При изложении материала авторы руководствовались также тем, чтобы он был полезен и инженерно-техническим работникам отрасли.

Учебное пособие написано в соответствии с типовыми и рабочими программами курсов, читаемых авторами студентам экономического и художественно-графического факультетов учреждения образования «Витебский государственный технологический университет».

Главы 1-4, 13 и разделы 5.1-5.4, 6.1-6.3, 7.1-7.2, 8.1-8.4, 9.1-9.6, 10.1-10.5, 11.1-11.3, 12.1-12.2 подготовлены Вардомацкой Е.Ю, разделы 5.5, 6.4, 7.3, 8.4, 9.7, 10.6, 11.4, 12.3 – Окишевой Т.Н.

Авторы выражают глубокую благодарность и признательность уважаемым рецензентам за ценные конструктивные замечания, способствующие повышению научно-методического уровня рукописи, а также редакционно-издательскому совету учреждения образования «Витебский государственный технологический университет», оказавшему неоценимую помощь и поддержку в процессе работы над пособием.

ГЛАВА 1. ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ

Обучение методам и средствам компьютерных информационных технологий – основа подготовки современного специалиста, способного принимать профессионально грамотные решения в стремительно изменяющемся мире XXI века. Без использования новейших инновационных технологий сегодня немыслимо ни одно производство.

В системе высшего образования на нее возлагаются задачи обучения приемам создания, хранения, воспроизводства, обработки и передачи данных средствами компьютерных технологий.

Информатика, как наука, призвана разрабатывать методологии создания информационного обеспечения процессов управления объектами на базе компьютерных информационных систем.

Информатика – компьютерная наука (Computer Science), находится на стыке нескольких наук — математики, физики, микроэлектроники, статистики, экономики. Это область человеческой деятельности, связанная с процессами преобразования информации с помощью компьютеров и их взаимодействием со средой применения [11].

Термин «информатика» возник в 60-х годах во Франции для названия области, занимающейся автоматизированной обработкой информации с помощью ЭВМ. Французский термин «информатика» образован путём слияния слов “information” (информация) и “automatigus” (автоматика) и означает «информационная автоматика или автоматизированная обработка информации». Выделение информатики как самостоятельной области человеческой деятельности в первую очередь связано с развитием компьютерной техники. Причем основная заслуга в этом процессе принадлежит микропроцессорной технике, появление которой в 70-х годах послужило началом второй электронной революции. Термин «информатика» теперь связывается не только с отображением достижений компьютерной техники, но и с процессами передачи и обработки информации. Информатика занимается изучением процессов преобразования и создания новой информации более широко, практически не решая задачу управления различными объектами, причем акцент ставится на свойствах информации и аппаратно-программных средствах её обработки.

Информация – это сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, которые уменьшают имеющуюся о них степень неопределённости, неполноты знания [4]. Информатика рассматривает информацию как концептуально связанные между собой сведения, данные, понятия, изменяющие наши представления о явлении или объекте окружающего мира. На рисунке 1.1 показана схема классификации информации по разным критериям.

Наряду с информацией, в информатике часто упоминается понятие «данные». Данные могут рассматриваться как признаки или записанные наблюдения, которые по каким-то причинам не используются, а только хранятся. В том

случае, если появляется возможность использовать эти данные для уменьшения неопределённости о чём-либо, данные превращаются в информацию. Поэтому можно утверждать, что информацией являются используемые данные.

Одной из важнейших разновидностей информации является *экономическая* информация. Её отличительная черта – связь с различными процессами управления. Экономическая информация сопровождает процессы производства, распределения, обмена и потребления материальных благ и услуг.

Экономическая информация – совокупность сведений, отражающих социально-экономические процессы, и служащих для управления этими процессами и коллективами людей в производственной и непроизводственной сферах. Информатика играет важную роль в современной экономической науке, что привело к выделению отдельного направления развития науки – экономической информатики. Это новое направление объединяет в себе экономику, математику и информатику и помогает экономистам решать задачи оптимизации деятельности предприятий, принимать стратегически важные решения о развитии промышленности и управлять производственным процессом. Разработанная программная база основывается на математических моделях экономических процессов и предоставляет гибкий и надежный механизм предсказания экономического эффекта управленческих решений. С помощью ЭВМ быстро решаются аналитические задачи, решение которых не под силу человеку. В последнее время компьютер стал неотъемлемой частью рабочего места управленца и экономиста.

На рисунке 1.1. приведена классификация информации по разным признакам

Рис 1.1. Классификация информации

Место возникновения

Входная информация – информация, поступающая в объект.

Выходная информация – информация, поступающая из объекта в другой объект.

Внутренняя информация возникает внутри объекта, внешняя – за пределами объекта.

Стабильность

Переменная (текущая) информация отражает фактические количественные и качественные характеристики производственно-хозяйственной деятельности.

Постоянная (условно-постоянная) информация – неизменная и многократно используемая в течение длительного периода времени информация. Она может быть:

- справочная (описание постоянных свойств объекта в виде устойчивых длительное время признаков);
- нормативная (местные, отраслевые и общегосударственные нормативы);
- плановая (многократно используемые в фирме плановые показатели).

Стадия обработки

Первичная информация – информация, которая возникает непосредственно в процессе деятельности объекта и регистрируется на начальной стадии.

Вторичная информация – информация, получаемая в результате обработки первичной. Она может быть промежуточной и результатной.

Промежуточная информация используется как исходные данные для последующих расчётов.

Результатная информация получается в процессе обработки первичной и промежуточной информации и используется для выработки управленческих решений.

Способ отображения

Текстовая информация – совокупность алфавитных, цифровых и специальных символов, с помощью которых представляется информация на физических носителях (бумага, дисплей и т.д.).

Графическая информация – различного рода графики, диаграммы, схемы, рисунки и т. д.

Функция управления

По функциям обычно классифицируют экономическую информацию.

Плановая информация – информация о параметрах объекта управления на будущий период.

Нормативно-справочная информация содержит различные нормативные и справочные данные. Её обновление происходит достаточно редко.

Учётная информация – информация, характеризующая деятельность фирмы за определённый прошлый период времени.

Оперативная (текущая) информация – информация, характеризующая производственные процессы в текущий (данный) период времени и используемая в оперативном управлении.

Создание и использование информационной системы для любой организации нацелены на решение следующих задач.

- Структура информационной системы, её функциональное назначение должны соответствовать целям, поставленным перед организацией.
- Информационная система должна контролироваться людьми, ими пониматься и использоваться в соответствии с основными социальными и этическими принципами.
- Производство достоверной, надёжной, своевременной и систематизированной информации.

Единицы измерения информации

Любую информацию компьютер переводит в так называемую бинарную или двоичную информацию. Она представляется всего двумя уровнями некоего сигнала – условным наличием его или отсутствием. Их называют логическая единица и нуль (1 и 0), ДА и НЕТ, True и False и т. д. Единица двоичной информации получила название бит. 8 бит = 1 байту – единице информации, имеющей 2^8=256 значений от 0 до 255.

256 значений байта хватает на кодирование достаточно большого числа знаков, например всех знаков русского или английского языков. Каждая последующая единица измерения объема информации, например килобайт, мегабайт и т. д., получается умножением предшествующей единицы не в 1000 раз, а в 1024 раза.

Вопросы для самоконтроля

1. Дайте определение и поясните понятие *«информация»*.
2. Дайте определение и поясните понятие *«информатика»*.
3. Дайте определение и поясните понятие *«экономическая информация»*.
4. Что такое *«данные»*, и чем они отличаются от информации.
5. По каким признакам может классифицироваться информация?
6. В каких единицах измеряется информация?
7. Охарактеризуйте виды постоянной информации.
8. Каким образом может отображаться информация?
9. Какие виды информации выделяют при классификации ее по стадиям обработки?
10. Какие виды информации можно выделить при классификации по функциям управления?

ТЕМА 1

ОСНОВНЫЕ СВЕДЕНИЯ О ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

ГЛАВА 2. ТЕХНИЧЕСКИЕ СРЕДСТВА ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

2.1. Состав и архитектура персонального компьютера

Персональный компьютер (ПК) – компьютер, предназначенный для индивидуального пользования. В минимальной конфигурации персональный компьютер состоит из следующих основных компонент: системного блока, клавиатуры, дисплея и манипулятора-мыши [11]. Системный блок содержит следующие устройства (рис. 2.1):

Рис.2.1. Архитектура персонального компьютера

Центральный процессор (системный блок), выполняющий вычисления в соответствии с заданной программой и управляющий работой компьютера. Центральный процессор содержит:

основной процессор – для выполнения логических и математических операций;

математический сопроцессор – для ускорения вычислений с плавающей точкой;

оперативную память (оперативное запоминающее устройство), в которой располагаются программы, выполняемые компьютером, и используемые программой данные. Для работы с современным программным обеспечением ёмкость оперативной памяти современных персональных компьютеров должна быть не менее 32 Мб;

постоянную память (постоянное запоминающее устройство), которая содержит базовую систему ввода-вывода (BIOS), управляющие программы и команды;

электронные схемы (контроллеры и адаптеры), управляющие работой различных устройств, входящих в компьютер;

порты ввода-вывода, через которые производится обмен данными с внешними устройствами. Имеются специализированные порты, через которые происходит обмен данными с внутренними устройствами компьютера, и порты общего назначения, к которым могут подсоединяться различные дополнительные внешние устройства (принтер, мышь, сетевой адаптер и т.п.). Порты общего назначения бывают двух видов: *параллельные*, обозначаемые LPT1-LPT4, и *асинхронные последовательные*, обозначаемые COM1-COM3. Параллельные порты выполняют ввод и вывод с большей скоростью, чем последовательные, но требуют и большего числа проводов для обмена данными;

блок дисков, состоящий из дисководов для гибких магнитных дисков (дискет), из дисководов для компакт-дисков (CD и DVD-дисков), накопителя на жестком магнитном диске (винчестера);

дисплей (монитор), предназначенный для отображения информации, в зависимости от адаптера, которым он комплектуется, может быть монохромным цифровым, монохромным графическим или цветным графическим.

Передача информации между устройствами компьютера, находящимися в системном блоке, осуществляется по *системной шине.*

Таким образом, архитектура персонального компьютера, которая изображена на рисунке 2.1, дает общее представление о входящих в его состав устройствах и функциональных взаимосвязях между ними.

2.2. Классификация персональных компьютеров

Главным фактором, определяющим вычислительные возможности персонального компьютера, является тип установленного в компьютер микропроцес-

сора. В зависимости от использованного микропроцессора различают следующие классы ПК:

- компьютеры класса XT, созданные на базе микропроцессора Intel 8080/86 (1978г.);
- компьютеры класса AT, созданные на базе микропроцессора Intel 80286 (1982г.);
- компьютеры класса 386, созданные на базе микропроцессора Intel 80386 (1985г.);
- компьютеры класса 486, созданные на базе микропроцессора Intel 80486 (1989г.);
- компьютеры класса Pentium, снабженные микропроцессором Pentium фирмы Intel (1995г.);
- компьютеры класса PentiumII, снабженные микропроцессором Pentium фирмы Intel (1998г.);
- компьютеры класса PentiumIII, снабженные микропроцессором Pentium фирмы Intel (1999г.);
- компьютеры класса Pentium4, снабженные микропроцессором Pentium фирмы Intel (2000г.).
- процессоры Intel Itanium™ (2001).
- процессоров Intel Xeon™ MP и Intel® Itanium 2 (2002).

Несмотря на значительные различия, ПК старшего семейства способны выполнять все программы, разработанные для младшего семейства.

В качестве перспективных можно рассматривать ПК класса Pentium, представители других классов морально устарели, и их выпуск прекращен.

В соответствии с целевым назначением компьютеры подразделяют на бытовые, учебные и профессиональные.

Бытовые – это наиболее простые, маломощные компьютеры, служащие для индивидуального применения в бытовых условиях (Celeron).

К учебным относятся компьютеры среднего класса, предназначенные для обучения различным предметам в учебных заведениях (ПК Pentium, Celeron на базе процессоров Intel, Cyrix).

Профессиональными являются мощные ПК, обеспечивающие удовлетворение интересов профессиональных пользователей

Кроме того, персональные компьютеры можно классифицировать следующим образом:

- *настольные* используются в стационарных условиях комнаты или кабинета и располагаются на рабочем столе;
- *переносные*, так называемые «лаптопы» (laptop - лежащий на коленях) или «ноутбуки» (notebook - блокнотный компьютер);
- *наладонные* – карманные переносные компьютеры. Они помещаются на ладони и их так и называют - наладонники или по-английски - палмтоп, что означает – лежащий на ладони.

2.3. Периферийные устройства персональных компьютеров

Внешние запоминающие устройства

Внешние запоминающие устройства (ВЗУ) обеспечивают долговременное хранение программ и данных на различных носителях информации. Наибольшее распространение получили накопители на магнитных и оптических дисках. Любой компьютер содержит как минимум один накопитель на гибких магнитных дисках и накопитель на жестких магнитных дисках.

Накопители на гибких магнитных дисках (НГМД) являются устройством со сменным носителем информации. Они обеспечивают резервирование информации, секретность данных и распространение программного обеспечения.

Накопитель на гибких магнитных дисках состоит из следующих узлов:

- механического привода, обеспечивающего вращение диска;
- блока магнитных головок чтения/записи;
- системы позиционирования магнитных головок, которая служит для их перемещения относительно диска в радиальном направлении;
- электронного блока, обеспечивающего управление накопителем.

Информация на магнитном диске размещается вдоль концентрических окружностей, называемых дорожками. Каждая дорожка содержит определенное число секторов. По номеру сектора и дорожки определяется местонахождение информации на диске. Наибольшее распространение получили двухсторонние, высокой плотности дискеты диаметром 3,5", емкостью 1,44 Мб.

Накопители на жестких магнитных дисках (НЖМД) с несъёмным носителем информации – магнитным диском (винчестером), системой позиционирования и блоком магнитных головок помещены в герметично закрытый корпус. Сам диск имеет металлическую основу. В НЖМД устанавливается несколько дисков суммарной емкостью в среднем до 512 Гбайт и даже выше. В отличие от гибкого, жесткий диск вращается непрерывно при включенном питании компьютера, имеет повышенную долговечность в связи с тем, что в нем отсутствует непосредственный контакт магнитных головок с поверхностью диска.

Накопители на оптических дисках (НОД) применяются все чаще в связи с широким распространением мультимедиа-приложений, интегрирующих тексты, звук и видео [8]. Принцип работы оптического диска основан на использовании луча лазера для записи/считывания информации в цифровом виде. В отличие от магнитного диска, оптический диск имеет одну спиральную дорожку и вращается в накопителе так, чтобы обеспечить постоянную линейную скорость, т.е. угловая скорость вращения диска увеличивается в процессе перемещения головки чтения к его центру.

CD-R (Compact Disk Recordable – **компакт диск записываемый**) – это лазерная оптическая система одноразовой записи. Она состоит из дисковода и записываемого компакт-диска. Диски CD-R представлены двумя видами: объемом 650 Мб и 700 Мб.

CD-RW (Compact Disk ReWritable – компакт диск перезаписываемый) – это оптическая система многоразовой записи. Она состоит из дисковода и перезаписываемого компакт-диска. Диски CD-RW также бывают двух объемов: 650 Мб и 700 Мб.

DVD (Digital Versatile Disc) - цифровой универсальный диск. В DVD в несколько раз увеличена плотность записи и уменьшены размеры наносимых меток.

Flash (карманный) – переносной носитель информации емкостью от 64 Мб до 8 Гб.

Устройства ввода-вывода информации

Основным устройством ввода информации в персональный компьютер является *клавиатура.* Она представляет собой матрицу клавиш, объединенных в единое целое, и электронный блок для преобразования нажатия клавиш в двоичный код. По своему назначению клавиши разбиваются на группы: алфавитно-цифровые, служебные, функциональные, клавиши малой цифровой клавиатуры и клавиши управления курсором.

Манипулятор «мышь» является дополнительным периферийным устройством ввода. Имеет не менее двух кнопок. Действия «мышью» выполняются с использованием левой и правой клавиш.

Дисплей (монитор) – это основное устройство, служащее для отображения выводимой на экран информации. Независимо от физических принципов формирования изображения дисплей состоит из двух основных частей – экрана и электронного блока, размещенных в одном корпусе. По функциональным возможностям дисплеи подразделяются на алфавитно-цифровые и графические. *Алфавитно-цифровые* способны воспроизводить только ограниченный набор символов. *Графические* в состоянии отображать как графическую, так и символьную информацию. По количеству воспроизводимых цветов различают *монохромные* и *цветные* дисплеи. По физическим принципам формирования изображения можно выделить мониторы на *базе электронно-лучевой* трубки ЭЛТ (CRT- Cathode Ray Tube) – мониторы и жидкокристаллические дисплеи, созданные на основе жидкокристаллических панелей, так называемыми ЖК (LCD - Liquid Crystal Display) – мониторы. Практически все компьютеры сейчас комплектуются мониторами Super VGA (SVGA). Возможности компьютера по отображению информации определяются совокупностью и совместимостью технических характеристик дисплея и его адаптера. Всякий адаптер содержит видеопамять, хранящую воспроизводимую на экране информацию. Каждой точке экрана соответствует поле видеопамяти, в котором хранится элемент изображения, определяющий режим высвечивания и цвет точки или символа. В настоящее время повсеместно используются видеоадаптеры типа Super VGA 32-х и 64-х битные с разрешением 1024x768 точек и выше: 1152x864, 1280x1024.

Печатающие устройства, или принтеры, предназначены для вывода алфавитно-цифровой или графической информации на бумагу. Наибольшее распространение получили *матричные, лазерные* и *струйные* принтеры.

С появлением цифровых фотоаппаратов, возникла необходимость использовать их не только для создания цифровых фотоизображений, но и для печати обычных бумажных фотографий. Для этой цели были разработаны *сублимационные* принтеры. Сублимационная технология печати ранее была применена в цветных копировальных аппаратах.

Основным узлом *матричного принтера* является печатающая головка, содержащая тонкие металлические иглы. Наиболее распространенные устройства имеют головку с 9 и 24 иглами и способны работать в двух режимах – текстовом и графическом. Скорость работы матричных принтеров достаточно низкая. В основе работы *лазерных* принтеров лежит электрографический принцип печати, заимствованный из ксерографии. С помощью лазерного луча изображение, подлежащее печати, проецируется на вращающийся барабан, покрытый светочувствительным материалом. После этого на барабан наносится порошок-тонер, частицы которого прилипают к барабану только в тех местах, которые были обработаны лазерным лучом. Путем прижатия бумаги к барабану на нее переносится и фиксируется рисунок. Изображение формируется по точкам, но за счет высокого разрешения лазерными принтерами обеспечивается качество, близкое к типографскому. *В струйных принтерах* формируется непрерывный поток из маленьких капель чернил, которые заряжаются и, пролетая через электрическое поле, отклоняются в вертикальной плоскости пропорционально их заряду. В цветных струйных принтерах печатающая головка имеет три сопла, которые формируют потоки капель красных, зеленых и синих чернил. Смешивая капли, можно получить на бумаге точку любого цвета. В *сублимационных* принтерах красящий порошок наносится также как в фотодиодных принтерах, но затем при помощи нагревательных элементов каждая частичка порошка очень быстро плавится и спекается. Получается четкое, яркое изображение. Печать ведется на бумагу, по составу похожую на обычную фотобумагу, но без желатинового слоя. Бумага для фото принтеров бывает матовой и глянцевой.

Кроме основных устройств ввода-вывода информации, существуют и дополнительные, имеющие специфическое назначение.

Плоттер (Plotter) или графопостроитель – устройство, позволяющее выводить на бумагу большого формата проектно-конструкторскую документацию: технические чертежи, плакаты, географические карты или любую графическую информацию. Плоттеры бывают монохромными и цветными. По технологии нанесения изображения плоттеры делятся на перьевые и струйные.

Дигитайзер (Digitizer) — устройство для оцифровки чертежей и других изображений. Дигитайзер обеспечивает считывание (скалывание) информации с листа вручную и позволяет преобразовать изображения в цифровую форму для обработки в компьютере.

Стриммер (stream — длинная лента) — устройство для записи информации на магнитную ленту. Стриммер используется для архивирования информации с жесткого диска.

Сканер (Scanner) —устройство для копирования информации, позволяющее с помощью оптики считывать графическую и текстовую информацию с листа в компьютер.

2.4. Вопросы для самоконтроля

1. Перечислите основные аппаратные средства ПК.
2. По каким признакам и как могут классифицироваться ПК?
3. Перечислите классы компьютеров, созданных на базе микропроцессоров Intel.
4. Какие компоненты расположены на материнской плате?
5. Для чего используется системная шина?
6. Как подразделяются порты ввода-вывода?
7. Перечислите и охарактеризуйте основные устройства ввода-вывода информации.
8. Какие группы можно условно выделить на клавиатуре?
9. Какие виды мониторов распространены в настоящее время?
10. Из каких узлов состоит накопитель на гибких магнитных дисках?
11. По каким критериям можно классифицировать мониторы?
12. Какие устройства предназначены для вывода алфавитно-цифровой или графической информации на бумагу?
13. Какой принцип печати заложен в основу работы матричного принтера?
14. Что лежит в основе работы лазерного принтера?
15. На чем основана работа струйного принтера?
16. Перечислите и охарактеризуйте современные внешние запоминающие устройства.
17. Перечислите и охарактеризуйте дополнительные устройства ввода-вывода информации.
18. Дайте сравнительную характеристику таким носителям информации, как CD-R, CD-RW, DVD.
19. Объясните, чем плоттер отличается от принтера.
20. Объясните, как происходит печать на сублимационном принтере.

ГЛАВА 3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Программное обеспечение любого персонального компьютера можно разделить на три основные части:

- операционная система и сервисные программы;
- инструментальные языки и системы программирования;
- прикладные программные средства.

3.1. Понятия, основные функции и составные части операционной системы

Операционная система (ОС) - это совокупность программ, содержащая основные инструкции для центрального процессора по управлению компьютером и всеми его элементами [7]. Являясь неотъемлемой частью современного компьютера, ОС обеспечивают поддержку работы программы с аппаратурой и предоставление пользователю возможностей общего управления компьютером. В настоящее время существует большое количество ОС для персональных компьютеров, отражающих как этапы развития технических средств, так и стремление разработчиков улучшить функциональные и эксплуатационные характеристики, повысить степень комфортности операционной системы по отношению к пользователю. По мере развития ОС выделились их основные функции:

- организация передачи информации между различными внутренними устройствами (например, между оперативной памятью и дисководом);
- обеспечение выполнения прикладных программ пользователя и системных программ;
- поддержка работы периферийных устройств (ввода/вывода);
- распределение ресурсов между задачами и поддержка распределения затрат при их параллельной обработке.

Перечисленные функции реализуются ядром операционной системы. Окружением ядра являются системные утилиты, редакторы, компиляторы и другие программные средства, составляющие обслуживающую часть операционной системы. Ядро обеспечивает базовые функции для окружающего программного обеспечения и допускает расширение обслуживающей части ОС. Например, поддержку СУБД, графический интерфейс, и т.д. Ядро ОС постоянно находится в памяти. Остальное программное обеспечение располагается на внешних запоминающих устройствах.

Важнейшими характеристиками, определяющими выбор операционной системы, являются следующие:

- распространенность;
- наличие большого количества прикладных программных средств, работающих под ее управлением;

- простота освоения и взаимодействия с ней пользователя;
- легкость перехода с одной версии ОС к другой, более совершенной.

Основными препятствиями к использованию прикладного программного обеспечения в различных операционных системах являются:

- разные способы организации файлов,
- несовместимость форматов дисков (различное число секторов и различная емкость).

В различных системах ПЭВМ используются системы с разной архитектурой и возможностями. Для их работы необходимы различные ресурсы оперативной памяти. Они предоставляют разные степени сервиса как для программ, так и для работы пользователей.

3.1.1. Операционная система MS DOS

Другой класс образуют ОС с более развитыми средствами доступа ко всем аппаратным компонентам, гибкой файловой системой, основанной на иерархической системе каталогов, удобным для пользователя командным языком. Средства, предоставляемые ОС этого класса, позволяют, с одной стороны, формировать удобную среду для разработки программного обеспечения (ПО), с другой стороны, на их основе можно разрабатывать автоматизированные рабочие места (АРМ) с простыми средствами доступа пользователей, научное и прикладное программное обеспечение.

MS DOS (Microsoft Disk Operation System) - это 16-разрядная однопользовательская однопроцессорная ОС, принятая в качестве базовой для работы на компьютерах IBM PC. Программы DOS работают и выполняются в пределах одного мегабайта адресного пространства компьютера, а основная память может использоваться для хранения данных. К ее основным достоинствам относятся:

- развитый командный язык;
- возможность организации многоуровневых каталогов;
- возможность работы со всеми пользовательскими устройствами, как с файлами;
- возможность подключения пользователем дополнительных внешних устройств.

Для MS DOS разработан большой арсенал программных средств. Имеются трансляторы практически для всех популярных алгоритмических языков высокого уровня, таких, как BASIC, PASCAL, C, FORTRAN, КОБОЛ, ЛИСП, ПРОЛОГ и др. Причем для большинства языков существует несколько вариантов трансляторов. Имеются инструментальные средства разработки программ в машинных кодах: ассемблеры, отладчики и др. Они включают в себя текстовые редакторы, компоновщики, редакторы связей и другие сервисные средства, необходимые для создания сложных программ.

Первая версия MS DOS разработана фирмой Microsoft в 1981 г. для компьютеров IBM в 1981 году, последняя версия MS DOS 6.2 – в 1994 году.

Важнейшей особенностью MS DOS является ее модульность, что позволяет при необходимости расширения функций системы модифицировать ее модули. В состав ОС входят следующие модули:

1. *Базовая система ввода-вывода (BIOS),* находящаяся в ПЗУ компьютера. BIOS содержит драйверы стандартных периферийных устройств, тестовые программы для контроля работоспособности оборудования и программу вызова системного загрузчика ОС.

2. *Системный загрузчик* размещается в первом секторе нулевой дорожки системного диска. Его основная функция заключается в считывании с диска в оперативную память модуля расширения BIOS и базового модуля.

3. *Модуль расширения BIOS* (файл IO.SYS) организует интерфейс с BIOS и подключает внешние драйверы периферийных устройств. Указания на подключение внешних драйверов должны содержаться в файле конфигурации CONFIG.SYS. Базовый модуль (файл MSDOS.SYS) реализует основные функции по управлению всеми ресурсами ПЭВМ и выполняемыми программами.

4. *Командный процессор* или интерпретатор команд (файл COMMAND.COM). Основные его функции заключаются в приеме, анализе, выполнении команд пользователя и обработке командных файлов (файлы типа BAT).

5. *Утилиты MS DOS* - это программы, поставляемые вместе с ОС в виде отдельных файлов с расширением COM или EXE. Они выполняют действие обслуживающего характера, например, форматирование дискет и т.п.

6. *Внешние, устанавливаемые драйверы периферийных устройств* - это специальные программы, дополняющие BIOS и обеспечивающие обслуживание новых ПУ или нестандартное использование имеющихся.

Одним из назначений DOS была работа с файловой системой. Файловая система MS DOS имеет *иерархическую* (древовидную) структуру.

Информация на дисках хранится в файлах. *Файл* - это именованная область на диске для хранения информации любого вида. В MS DOS файл характеризуется составным именем, атрибутами, датой и временем создания, длиной файла в байтах.

Составное (полное) имя файла складывается из имени файла и его расширения, характеризующего содержимое файла.

Атрибуты файла определяют способы его использования и права доступа к нему. ОС допускает задание следующих атрибутов:

- R - файл предназначен только для чтения, его нельзя удалить и изменить с помощью команд MS DOS;
- A - архивный файл, если его изменять, то будет создана архивная копия;
- H - скрытый файл, доступ к которому обычным способом невозможен;
- S - системный файл, Это скрытый, доступный только для чтения файл, используемый MS DOS. Он не поддается модификации или удалению.

Под *файловой структурой* понимают совокупность файлов и взаимосвязей между ними. Файловая система MS DOS позволяет объединить файлы в каталоги. *Каталогом* называется специальный файл, в котором регистрируются другие файлы и каталоги. Так образуется разветвленная (древовидная) файловая структура на диске (рис. 3.1).

На каждом диске всегда имеется единственный корневой каталог, создаваемый при его форматировании. В корневой каталог входят файлы и подкаталоги 1-го уровня (NC, E_83, EXE), в подкаталоги 1-го уровня - файлы и подкаталоги 2-го уровня (Nc.exe, Nc.mnu, Be.exe, Primer.pas и т.д.).

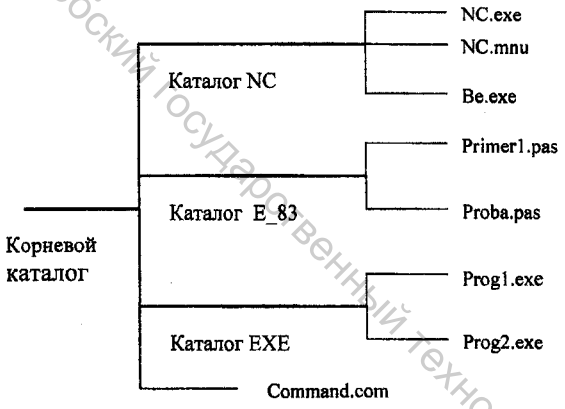


Рис.3.1. Дерево каталогов

Каталог, с которым в настоящий момент работает пользователь, называется *рабочим*. Его имя обычно указывается последним в строке приглашения MS DOS.

Если в команде ОС используется файл не из рабочего каталога, необходимо указывать путь к файлу. *Путь* состоит из последовательности имен каталогов или символов «..», разделенных символом «\» (слэш). Для обеспечения доступа к файлу в общем случае необходимо задать его спецификацию:

```
[имя_диска:\] [путь\]имя_файла.
```

Например, `d:\E_83\Proba.pas`.

Кроме системного ПО, для MS DOS создано множество прикладных программ, которые могут работать на различных ПЭВМ, где установлена эта операционная система.

Одно из направлений развития MS DOS связано с созданием особых программных надстроек, называемых *операционными оболочками*, которые обеспечивают удобный интерфейс пользователя с ОС и некоторые другие дополнительные функции. К популярным программным надстройкам MS DOS относятся Norton Commander, PCToolls, Volkov Commander и некоторые другие. Они дают возможность компактно размещать на накопителе магнитных дисков необходи-

мые для работы программы и данные, очищать диски от устаревших записей, быстро находить требуемую информацию, исправлять и дополнять ее, а также выполнять многие другие функции DOS без обращения к резидентным командам и утилитам ОС.

Другое направление развития ОС связано с поддержкой новых технических средств, в частности, с расширением до нескольких десятков и более мегабайт ОП, новых видов накопителей, видеомониторов и пр. Наиболее популярной операционной оболочкой, запускаемой на выполнение как обычная программа MS DOS, является WINDOWS 3.1, разработанная в 1991 г., и позже WINDOWS 3.11, как сетевой вариант.

3.1.2. Семейство операционных систем Windows

Операционная система Windows 3.1 представляет собой однопользовательскую операционную систему, работающую на базе MS DOS и использующую на нижнем уровне внутренние функции и процедуры этой ОС. Принципиальным условием для программных приложений, предназначенных для работы в среде Windows 3.1, является то, что они должны работать с внешними устройствами (монитор, принтер) не напрямую, а через универсальную систему команд. Управляющая система транслирует вызовы (обращение к тому или иному физическому устройству) и передает их соответствующему драйверу данного устройства, который непосредственно отвечает за работу с ними. Основу пользовательского интерфейса Windows 3.1 составляют иерархически организованная система окон и другие графические объекты. Пользовательский интерфейс Windows, в отличие от интерфейса командной строки MS DOS и оболочки NC, реализует оперативное управление на основе выбора того или иного графически визуализированного элемента (кнопки, пиктограммы и диалогового окна, мастера подсказок и др.) с помощью мыши, а команды клавиатуры и «горячие клавиши» имеют резервное значение. Сетевая версия Windows 3.1 – Windows 3.11.

Операционная система Windows 95 является высокопроизводительной, многозадачной, многопотоковой 32-разрядной операционной системой с графическим интерфейсом и расширенными сетевыми возможностями, работающей в защищенном режиме, поддерживающей 16-разрядные приложения без всякой их модификации. Эта интегрированная среда обеспечивает эффективный обмен текстовой, графической, звуковой и видеоинформацией между различными программами. Функциональные возможности Windows 95 качественно превосходят MS DOS, Windows 3.1, Windows 3.11. Windows 95 - это полномасштабная информационная система семейства Windows, не требующая поддержки MS DOS. Она полностью совместима с используемыми в настоящее время аппаратными и программными средствами. Windows 95 - это первый представитель нового поколения 32-битовых ОС.

К преимуществам ОС Windows 95 можно отнести следующие [2]:

- *интегрированная ОС,* ядро которой загружается в момент включения компьютера, активизируя графический интерфейс и обеспечивая полную совместимость с ОС MS DOS;
- *вытесняющая многозадачность* - свойство ОС самостоятельно в зависимости от внутренней ситуации передавать или забирать управление того или иного приложения, не позволяя одному приложению забирать все аппаратные ресурсы;
- *многопоточность* - свойство ОС выполнять операции одновременно над потоками нескольких 32-битных приложений, называемых процессами.

Windows 95 использует технологию Plug and Play, т.е. упрощает работу с компьютером за счет следующих функций:

- помощь при распознавании устройств для их установки и настройки;
- динамическое изменение состояния системы и автоматическое уведомление об этом программных приложений;
- интеграция драйверов устройств системных компонентов и пользовательского интерфейса.

Windows 95 обеспечивает динамическое изменение конфигурации системы, упрощает настройку устройств и управление оборудованием. Специальная программа *диспетчер устройств* позволяет получать информацию обо всех найденных системой устройствах и изменять при необходимости их конфигурацию. Windows 95 - это высокоэффективная платформа для мультимедиа, которая включает в себя CD или лазерный проигрыватель, обеспечивает поддержку видеодисков и видеомагнитофонов, кроме того, имеет широкие возможности для пользователей с ограниченными возможностями:

- возможность масштабирования элементов интерфейса;
- запоминающие клавиши;
- режим Mouse Keys (все действия с мышью можно выполнить через клавиатуру);
- визуальное дублирование звуков системы.

Операционная система Windows 98 по сравнению с Windows 95 включает средства, позволяющие компьютеру работать быстрее без добавления нового оборудования. В состав Windows 98 входит ряд программ, совместное применение которых повышает производительность компьютера.

В Windows 98 надежность повышается за счет того, что применяются новые мастера. Например, *Мастер обслуживания* позволяет быстрее выполнять программы, проверять жесткий диск на наличие ошибок и освобождать место на диске. Кроме того, служебные программы и ресурсы обеспечивают бесперебойную работу системы, поддерживая перечисленные ниже функции.

Проверка системных файлов позволяет отслеживать состояние наиболее важных файлов, обеспечивающих работу компьютера. Если эти файлы повреждены или перемещены, программа проверки их восстанавливает.

Проверка диска производится автоматически после неверного выключения ОС. Программа проверки дисков обнаруживает наиболее вероятные повреждения дисков и исправляет ошибки. Пользователь имеет возможность выполнить проверку диска в любой момент.

Новый Web-узел ресурсов MS Windows Update автоматизирует процесс обновления драйверов и системных файлов и обеспечивает новейшие возможности технической поддержки.

Системная программа «проверка реестра» позволяет обслуживать и проверять реестр на наличие ошибок и несогласованности структур данных при каждом запуске компьютера.

Программы архивации предоставляют расширенные возможности архивации и восстановления данных.

Использование мастеров специальных возможностей позволяет пользоваться ПК людям с физическими недостатками.

Обозреватель Интернета Internet Explorer позволяет сделать ряд функций доступными с рабочего стола Windows:

- возможность поиска в сети из любого места на компьютере;
- возможность визуализации канала Web-узлов на рабочем столе;
- подписку на избранные узлы;
- настройку панели ссылок;
- настройку панели обозревателя;
- контроль содержимого из зоны безопасности при поиске Web.

Проводник Windows и Internet Explorer позволяют объединить ресурсы Windows в едином представлении. Windows 98 делает наиболее продуктивным использование Web за счет применения всех возможностей ПК к интерактивному содержимому Internet:

- автоматического добавления ранее вызывавшихся адресов web по мере их ввода;
- улучшенных списков часто посещаемых web-узлов;
- улучшенного журнала и возможности обследования часто посещаемых web-узлов;
- поддержки всех основных стандартов Интернета, в т.ч. и Java;
- высокой производительности динамических документов HTML, что позволяет сделать web-страницы разнообразными и интересными;
- нового мастера подключений к Internet, позволяющего зарегистрироваться для доступа к Интернет и автоматически выполнить шаги по настройке программного обеспечения для доступа к сети.

Приложение Inernet Explorer объединяет рабочий стол с web, благодаря чему рабочий стол и его папки будут выглядеть и действовать так же, как и при работе с сетью. Аналогично просмотру содержимого web можно просматривать и содержимое компьютера, и при этом просмотр web возможен из любого места компьютера. Кроме того, на рабочий стол и на панель задач можно добавить активное содержимое, такое, как, например, бегущая строка новостей.

Операционная система Windows 2000 по сравнению с ранее разработанными версиями поддерживает ряд новых технических решений:

1. Защита данных средствами *протокола аутентификации*, обеспечивающего передачу данных незащищенным путем. Именно он, наряду со службой каталогов Active Directory, позволяет обеспечивать важнейшую особенность Windows 2000 по сравнению с Windows NT и предоставляет ряд преимуществ по сравнению с HTML-протоколами аутентификаций.

2. *Инфраструктура открытых ключей.* ОС Windows 2000 оснащается средствами защиты информации на базе структуры открытых ключей. Эта структура представляет собой систему цифровых сертификатов и организаций, удостоверяющих сертификаты CAS. Так же, как и система Kerberous, она дает возможность обоим участникам транзакции (участникам обмена сообщениями) проверить, действительно ли партнер является тем, за кого себя выдает, а также шифровать транзакции. Система защиты данных на базе структуры открытых ключей лучше всего подходит для использования в сети Интернет и является полезным средством для организации электронных конференций между предпринимателями, для предоставления клиентам круглосуточного доступа по каналам Интернет. Таким образом, уровни безопасности и шифрования данных станут неотъемлемой частью в повседневной коммерческой практике.

3. Windows 2000 оснащена рядом средств, предназначенных для пользователей малых офисов. В состав системы входит качественный *маршрутизатор*, обладающий интерфейсом подключения по запросу, что облегчает процесс установки соединения с интернет-провайдером. При помощи таких средств, как реализованный в Windows 2000 сервер-доступ, к одному предоставленному выходу в Интернет направляется сразу несколько пользователей сети. В данном модуле реализована технология трансляции сетевых адресов NAT, которая дает возможность компьютерам, объединенным в единую локальную Windows-сеть выходить через маршрутизатор в Интернет, используя единственный IP-адрес от Internet-провайдера. Множество пользователей локальной сети являются, с точки зрения провайдера, единым клиентом.

4. Windows 2000 имеет встроенные *средства удаленного доступа.*

5. Новые функции управления хранением данных: управление квотами дискового пространства, иерархическая система хранения данных, динамическое управление ими, а также путем внесения изменений в файловую систему NTFS.

6. *Дополнительные возможности для мобильных пользователей,* доступ к сетевым файлам и папкам для работы с ними без подключения к сети. ОС создает локальный КЭШ, сохраняет в нем файлы и папки, и они синхронизируются с оригиналами в фоновом режиме. В Windows 2000 предусмотрена функция перехода в режим «спячки». Перед отключением питания все содержимое физической памяти записывается в специальный файл.

7. *Расширенное административное управление ОС*, что позволяет снизить общую стоимость эксплуатации ОС.

Операционная система Windows XP официально была представлена 25 октября 2001 года. Она заменила на прилавках Windows 2000, которая так и не обрела большой популярности.

Windows XP - система, позволяющая работать в терминальном режиме, имеющая возможность одновременной работы неограниченного количества пользователей при добавлении утилиты Desktop manager и позволяющая работать одновременно на четырех рабочих столах. Windows XP выпущена в трех версиях: XP Home Edition, XP Professional и XP 64-Bit Edition.

Операционная система *XP Home Edition,* предназначена для домашнего использования и малого бизнеса, по сути является модификацией Windows 9x.

Операционная система *XP Professional* предназначена для пользователей Windows 2000 или Windows NT. Хотя XP Home Edition и XP Professional базируются на одном и том же ядре, ОС XP Professional, ориентированная на применение в сфере бизнеса, обладает более расширенной функциональностью по сравнению с домашней системой. Но требования и для XP Pro, и для XP Home примерно похожи.

В ОС Windows XP реализованы следующие новые технические решения:

1. Windows XP построена на усовершенствованном ядре Windows 2000 и имеет новый, простой интерфейс, который упрощает работу с компьютером.

2. Windows XP имеет *службу терминалов* Remote Assistance, которая позволяет дистанционно подключаться к компьютеру. Таким образом, можно получить доступ к любым документам и файлам на удаленной машине.

3. Создан *откат драйвера* (Rollback Driver), который препятствует сбою системы.

4. Добавлена функция «System Restore» (восстановление системы). С ее помощью пользователь создает системную метку, ОС запоминает конфигурацию системы, а затем, в случае неполадок, можно вернуть систему на некоторое время назад, когда всё было в порядке.

5. Windows XP позволяет записывать компакт-диски без привлечения посторонних программ. Нажатием одной кнопки файл может легко быть записан на диск.

6. Имеется *встроенный архиватор*, типа Zip.

7. В систему WindowsXP входит новый обозреватель Internet Explorer версии 6.0, а также почтовая программа Outlook Express 6. Версия Direct-8.

8. Брандмауэр интернет-подключений автоматически защищает подключенный к интернету ПК от несанкционированного доступа.

9. Проигрыватель Windows Media для Windows XP— полнофункциональное средство, обеспечивающее поиск, воспроизведение, упорядочивание и хранение цифрового мультимедиа-материала.

10.Мастер установки сети помогает легко подключать и совместно использовать компьютеры и устройства, применяемые в домашних условиях.

11.Служба сообщений Windows Messenger – эффективное средство связи и совместной работы, поддерживающее передачу сообщений, проведение голосовых и видеоконференций, а также совместное использование приложений.

12.Центр справки и поддержки упрощает решение текущих проблем и помогает своевременно получать необходимую техническую поддержку.

13.Беспроводное подключение – автоматическая беспроводная конфигурация сети с использованием стандарта 802.1x.

14.Автономные файлы и папки — доступ к файлам и папкам, хранящимся на общем сетевом диске даже во время отключения компьютера от сервера.

15.Шифрованная файловая система — защита важных данных, содержащихся в файлах, хранящихся на диске, на котором используется файловая система NTFS.

16.Управление доступом — запрещение доступа к избранным файлам, приложениям или другим ресурсам.

17.Отображение текста на разных языках (технология Single Worldwide Binary) — можно вводить текст на любом языке и запускать версию приложений Win32 для любого языка, используя соответствующую версию операционной системы Windows XP.

18.Многоязычный пользовательский интерфейс позволяет менять язык пользовательского интерфейса, чтобы работать с локализованными диалоговыми окнами, меню, файлами справки, словарями, средствами проверки правописания и т.д.

Операционная система Windows NT – это сетевая операционная система, предоставляющая возможность работы каждому пользователю в индивидуальной области, защищенной паролем, с распределенным доступом к сетевым ресурсам. Эта операционная система существует только для аппаратных платформ Alpha и Intel. Широкого распространения она не получила и в настоящее время используется ограниченно.

3.1.3. Семейство операционных систем Unix

Семейство *ОС Unix* является альтернативой семейству Windows. Основное отличие и преимущество этой операционной системы заключается в том, что она может использоваться для широкого круга аппаратных платформ. Начиная с момента своего появления в 1969 году, ОС Unix получила широкое распространение на машинах различной мощности и архитектуры. Эта ОС является не только многозадачной, но и многопользовательской системой, которая позволяет нескольким пользователям разделить вычислительные ресурсы одного компьютера. В первых версиях Unix взаимодействие с пользователем осуществлялось с помощью командной строки, затем появились варианты графического интерфейса, позволяющие существенно облегчить работу и сделать систему легкодоступной. В настоящее время существует много версий Unix от различных производителей. Наиболее известные версии Unix: SUN OS, Solaris для компьютеров SUN, ATX, IRIX – для IBM, SCON IX – для SCON. Эти ОС созданы для платформы Intel.

В настоящее время все большую популярность приобретают версии Unix для персональных компьютеров. Одной из них является ОС Linux, поддержи-

вающая большинство свойств, присущих ОС Unix. К особенностям Linux можно отнести следующие:

- поддержка национальной и стандартной клавиатур;
- динамически загружаемые драйверы;
- поддержка различных версий файловых систем;
- реализация файловой системы MS DOS, позволяющей прямо обращаться к файлам MS DOS на жестком диске;
- поддержка полного набора протоколов TCP/IP для работы в сети.

3.2. Сервисные программы и системы обслуживания

Сервисные программы – это вспомогательные инструменты, расширяющие и дополняющие функциональные возможности ОС. К сервисным программным средствам относятся:

- программы обслуживания диска;
- программы архивации;
- антивирусные средства.

3.2.1. Программы обслуживания диска

Форматирование диска. Накопители на магнитных носителях, перед тем как их можно будет использовать в качестве носителей информации должны пройти специальное форматирование. Для этого обычно используется стандартная программа Windows, которая доступна при вызове контекстного меню соответствующего устройства. Можно форматировать диски Floppy устройств, жесткие диски и диски других устройств, допускающих эту процедуру. Исключение составляют системные устройства, с которых была проведена загрузка системы. При форматировании диска информация, записанная на диске, теряется. Полное форматирование целесообразно применять для дисков, на которых появились сбойные участки.

Дефрагментация диска. Эта процедура связана с особенностью использования системы FAT. Доступ к файлу, расположенному в одном месте диска (файл размещается последовательно в смежных кластерах), занимает меньше времени, чем доступ к файлу, фрагменты которого разбросаны по всему диску. Чем больше фрагментов, тем меньше доступ к ним. Для увеличения фрагментов системы диск необходимо периодически дефрагментировать.

Проверка диска на наличие ошибок. В процессе эксплуатации диска возможно появление ошибок, связанное со сбоями в процессе записи на него. Целостность файловой системы обычно определяется:

- правильностью имен файлов (файлы с неправильными именами нельзя открыть);
- правильностью даты и времени создания файла (неправильная дата и

время могут влиять на работу многих программ, обрабатывающих такие файлы);

- уникальностью имен файлов;
- отсутствием мастеров, не принадлежащих ни одному файлу;
- отсутствием файлов с общими кластерами.

Проверка поверхности диска на наличие повреждений и аппаратных ошибок занимает больше времени, чем проверка целостности файловой системы. Для сокращения времени работы программы можно выбрать проверку не всего диска, а только системной области или области данных. При нахождении поврежденных секторов их данные обычно переносятся в другие сектора. Некоторые системные файлы нельзя переносить, поэтому рекомендуется исправление ошибок в секторах, в которых располагаются такие файлы, отменить. Для большей надежности проверки поверхности диска необходимо использовать режим записи на диск, при котором данные на диске не теряются. Для исправления других ошибок необходимо использовать другие программы, например, Norton Utilities.

Очистка диска. В процессе эксплуатации дисковых носителей свободное пространство заполняется файлами. Для создания новых файлов необходимо освободить место, занимаемое уже ненужными файлами, поэтому они удаляются. Удаленные файлы помещаются в корзину на рабочем столе. Они меняют место расположения, но не удаляются с диска. Для освобождения места, занятого файлами, необходимо очистить корзину на рабочем столе.

3.2.2. Программы архивации

Наиболее простым и эффективным способом восстановления испорченных данных на компьютере является создание копий. Для уменьшения места, необходимого для содержания копий и удобства эксплуатации, создано большое количество программ, позволяющих работать с архивными файлами. Они могут содержать один или более файлов в сжатом виде. Можно как извлекать из этого файла исходные данные, так и добавлять новые. Наиболее популярные программы архивации под Windows: WINZIP, WINRAR, WINAZJ, NETZIP.

3.2.3. Компьютерные вирусы и антивирусные средства

Вирус – это, как правило, небольшая по объему последовательность программных кодов, обладающих следующими свойствами:

- возможностью создавать свои копии и внедрять их в другие программные объекты;
- обеспечением скрытности до определенного момента;
- несанкционированностью (со стороны пользователя) производимых ею действий;

- наличием отрицательных последствий от ее функционирования.

Не все вирусы обладают всеми из свойств, но программы, которые обладают одним или некоторыми свойствами, могут и не являться вирусами. К причинам создания и развития индустрии вирусов можно отнести причины технического (ранние версии операционных систем), экономического (борьба с конкурентами) и субъективного (компьютерное хулиганство) характера.

Классификация вирусов:

- загрузочные вирусы;
- файловые вирусы;
- макровирусы;
- сетевые вирусы.

Загрузочные вирусы – внедряются в так называемые загрузочные области носителя. Эти области – это сектора накопительных устройств, предназначенные для хранения специальных программ, загрузчика ОС (если заражен системный диск, с которого происходит загрузка системы, то управление получит не обычная программа загрузки, а защищающая ее код от вируса). При заражении вирус считывает необходимую информацию с первоначального загрузчика и сохраняет ее в своих кодах. Как правило, загрузочные вирусы являются резидентными, т.е. при загрузке системы вирус устанавливается в память компьютера и находится там постоянно, перехватывая необходимое прерывание и производя все действия, на которые он запрограммирован.

Файловый вирус – использует особенности организации системы. Такие вирусы внедряются в выполняемые файлы, библиотеки и объектные модули, загружаемые драйвера, файлы исходных текстов программы. Файловые вирусы внедряются в начало, конец или середину файла. Такие файлы остаются работоспособными, так как код вируса дописывается к исходному коду, т.е. некоторые вирусы, вместо исходного содержимого файла, записывают свой код. Файловые вирусы могут и не изменять содержимое зараженных файлов. Иногда для своей активизации они используют свойство ОС, определяющее порядок запуска программы. Так, если в одном каталоге есть файлы с одинаковыми именами, то сначала запускается файл с расширением *.BAT, потом *.COM и *.EXE. Следовательно, если зараженный файл имеет расширение EXE, то вирус файлы с тем же именем, но с расширением COM будет запускать раньше.

Макровирусы – это файловые вирусы, использующие особенности файлов, документов, популярных редакторов и элементов таблиц. В этих файлах размещаются программы на макроядрах, возможности которых позволяют создавать программу-вирус.

Сетевые вирусы – используют для своего распространения возможности компьютерных сетей. Первоначально использовали ошибки в сетевых протоколах и программном обеспечении глобальных сетей. В настоящее время эти ошибки исправлены, и такие вирусы не имеют распространения. Сетевые вирусы распространяются, используя возможности электронной почты, в основном это макровирусы.

Резидентные вирусы – могут оставлять свои копии в оперативной памяти. Они остаются активными и после завершения работы программы (и даже после удаления файла с вирусом, вплоть до перезагрузки системы). Заражение компьютера происходит через устройства, позволяющие вводить информацию в компьютер: модем, сетевую карту, устройства хранения информации со сменными носителями.

Одним из основных последствий деятельности вирусов является потеря или порча информации, поэтому для обеспечения устойчивой и надежной работы желательно всегда иметь эталонные (незараженные, чистые) копии используемой информации и программного обеспечения. При заражении вирусом нарушается целостность информации. Использование специальных утилит, которые позволяют отслеживать информацию о системных отличиях и файлах (контролировать размеры, дату создания и др.), относительно быстро позволяет обнаружить проникновение вирусов. Использование специальных антивирусных средств в большинстве случаев позволяет не только определить вирусное вторжение, но и оперативно обезвредить обнаруженные вирусы и восстановить испорченную информацию. Популярные *антивирусные* эффективные программные системы: антивирус Касперского (AVP), Norton Antivirus, Doctor Weber (DRW).

К сожалению, не существует антивирусов, которые бы обеспечили полную защиту от вирусов. При выборе антивирусного обеспечения следует принимать во внимание перечень и качество предоставляемых услуг:

- возможность сканирования «на лету», когда любой объект, к которому осуществляется доступ, проверяется на наличие вируса;
- возможность сканирования по запросу, когда время и область действия антивируса определяются пользователем;
- возможность настройки антивируса в соответствии с потребностями пользователя.

Качество работы антивируса определяется его надежностью (отсутствие ошибок, зависания) и эффективностью (обезвреживание всех вирусов), отсутствием ложных срабатываний. Все антивирусные программы обновляются по мере появления новых вирусов, в том числе через Internet.

3.3. Системы программирования

Системы программирования – это языки программирования высокого уровня, машинно-ориентированные языки и инструментальные средства.

Машинно-ориентированные языки - это языки, более понятные ЭВМ. Среди наиболее распространенных машинно-ориентированных языков прежде всего можно выделить Ассемблер. Между тем, если посмотреть, что происходит в мире современных компьютеров, нетрудно заметить, что почти все программное обеспечение создается на языках высокого уровня, более понятных для человека.

Программу, написанную на этих языках, может разобрать не только ее автор, а в ряде случаев – даже квалифицированный пользователь.

Языки программирования высокого уровня разделяются на несколько основных групп или семейств.

В первую группу можно объединить все *алгоритмические языки,* которые позволяют заложить в программу способ решения - алгоритм. Наиболее известные представители этой самой большой семьи - Фортран, Алгол, Ада, Бейсик, Паскаль, Модула-2, Delphi, Си.

Фортран - формульный транслятор. Предназначен для решения научно-технических задач. Разработан фирмой IBM в середине 50-х годов. Недостатком Фортрана является то, что он плохо поддается структурированию, т.е. программу нельзя представить в виде отдельных модулей.

Бейсик - универсальный символический код инструкций для начинающих. Прямой потомок Фортрана. Появился в 1964 году. Занимает в памяти мало места. В отличие от Фортрана, Бейсик умеет обрабатывать не только числовые, но и символьные переменные, поэтому этот язык удобен для решения большинства примитивных повседневных задач. Разновидности: TurboBasic, GWBasic, Visual Basic for Applications.

Паскаль (назван так в честь знаменитого математика Блеза Паскаля) проигрывает в популярности только Бейсику. Язык истинно структурного программирования. Это значит, что программу на Паскале можно формировать из отдельных, совершенно независимых процедур и функций, каждая из которых призвана выполнять определенную ограниченную задачу. При таком подходе становится возможным быстрое создание больших программных комплексов. Прямые наследники Паскаля - Модула-2 и ПЛ/1, Delphi в которых учтены все слабые места Паскаля.

Delphi – система программирования на языке Паскаль, включающая в себя средства для формирования дружественного пользователю интерфейса: окон, элементов управления и т.п. С помощью Delphi можно разрабатывать приложения для Windows.

C, C++ задуманы как инструмент для реализации и развития известной операционной системы Unix. Возникли в начале 70-х годов. Являются языками системного программирования и позволяют решать такие задачи, которые в ином случае потребовали бы машинно-ориентированного языка Ассемблера. Привлекательной чертой C и C++ является наличие всех особенностей, присущих современному универсальному языку: структурности, модульности, определения типов данных, рекурсивности. Для начинающих программистов эти языки достаточно сложны.

Вторая группа - это *логические функциональные* языки, которые привлекают сейчас всеобщее внимание в связи с активными исследованиями в области искусственного интеллекта. Эта семья невелика, в качестве представителей можно назвать Лисп, Пролог.

Лисп (от англ. lisp - шорох, лепет), в отличие от процедурных языков типа Паскаля, является языком функционального программирования, ориентирован-

ным на динамическую обработку данных. Эффективен для работы со списками, может быть использован для задач, где отсутствует четкий алгоритм решения.

Пролог (программирование в логике) является европейским противовес американскому Лиспу. Относится к языкам для создания систем искусственного интеллекта, т.е. компьютер должен самостоятельно выводить желаемый результат из фактов и правил, не получая от программиста путь решения. Программист лишь определяет объекты и логические связи между ними.

3.4. Прикладные программные средства

Прикладные программные средства (ППС) составляют категорию программных средств, обращенных непосредственно к пользователю. Их назначение состоит в том, чтобы пользователь с помощью компьютера мог решать свои повседневные задачи, приобретал определенные навыки, проводил свой досуг.

К основным прикладным программным средствам можно отнести:

- *текстовые редакторы* (процессоры), *настольные издательские системы* позволяющие наиболее удобным образом создавать, корректировать текстовую и графическую информацию (Word, Page Maker, Page Plus, Corel Ventura, Quark X, MS Publisher, Press);
- *графические редакторы* – это пакеты, предназначенные для обработки графической информации. Различают растровые графические редакторы, рисующие изображение по точкам, причем для каждой точки задается цвет (Aldus Photo, Adobe Photo Shop, Photo Finish, Picture Man, Paintbrush) и векторные, рисующие изображение линиями (Corel Draw, Adobe Illustrator, MS Draw);
- *демонстрационная графика* – конструкторы графических образов деловой информации, предназначенные для представления в наглядной и динамичной форме результатов аналитических исследований. Эти пакеты снабжены дружественным интерфейсом и достаточно просты в работе (Power Point, World Perfect Presentation, Frulance Graphics);
- *пакеты программ мультимедиа* предназначены для использования ЭВМ для отображения и отработки аудио и видео информации. Помимо программных средств компьютер должен быть оснащен дополнительными платами, позволяющими осуществлять ввод-вывод аналоговой информации и ее преобразование в цифровую форму. В качестве носителей такой информации используются CD и DVD – диски;
- *табличные процессоры* (электронные таблицы), обеспечивающие удобные средства работы с двумерными таблицами (Lotus, SuperCalc, Excel);
- *системы управления базами данных* (СУБД), которые обеспечивают эффективные действия по обработке больших объемов информации, их поиску, корректировке и хранению. СУБД могут иметь свой язык программирования (Dbase-подобные СУБД, FoxPro, MS Access);
- *пакеты прикладных программ*, содержащие стандартные подпрограммы, для производства расчетов по известным алгоритмам;

- *прикладное программное обеспечение*, используемое для автоматизации работ в конкретной области (финансовые программы для ведения личных финансов, автоматизации бухучета малых и крупных фирм, разработки технико-экономических обоснований финансовых сделок – MS Money, Money Counts, Бемби, 1С Бухгалтерия система управления предприятием – автоматизированный комплекс «Галактика», САПР - AutoCAD);
- *обучающие* системы в различных областях знаний (энциклопедии).
- *игры* развлекательного и тренировочного характера;
- *вирусы* - программы, мешающие работе ПЭВМ, или разрушающие информацию и оборудование компьютера; они используются для защиты запатентованных программ или систем, нанесения информационного ущерба;
- *антивирусы* - программы для обнаружения и обезвреживания вирусов.

Могут использоваться также интегрированные системы, объединяющие наиболее часто используемые прикладные программы. Но любая прикладная программа может работать только под управлением определенной операционной системы.

3.5. Вопросы для самоконтроля

1. Что такое операционная система (ОС)?
2. Перечислите и поясните функции операционной системы.
3. Перечислите и поясните характеристики операционной системы.
4. Охарактеризуйте особенности операционной системы MS DOS
5. Какие программные средства могут работать под управлением ОС MS DOS?
6. Перечислите и охарактеризуйте модули ОС MS DOS.
7. Перечислите и охарактеризуйте функции ОС MS DOS.
8. Что собой представляет полное имя файла, путь к файлу?
9. Охарактеризуйте особенности ОС Windows 3.X.
10. Охарактеризуйте особенности ОС Windows 95.
11. Перечислите преимущества ОС Windows 95.
12. Какие новые технические решения использованы в ОС Windows 2000?
13. Поясните различия между Windows 2000 и ОС Windows XP.
14. Охарактеризуйте программы обслуживания жесткого диска.
15. Что такое «компьютерный вирус»?
16. Как классифицируются компьютерные вирусы?
17. Какие существуют средства защиты от вирусов?
18. Какие языки можно отнести к машинно-ориентированным?
19. Перечислите и охарактеризуйте языки программирования высокого уровня.
20. Как классифицируются прикладные программные средства?

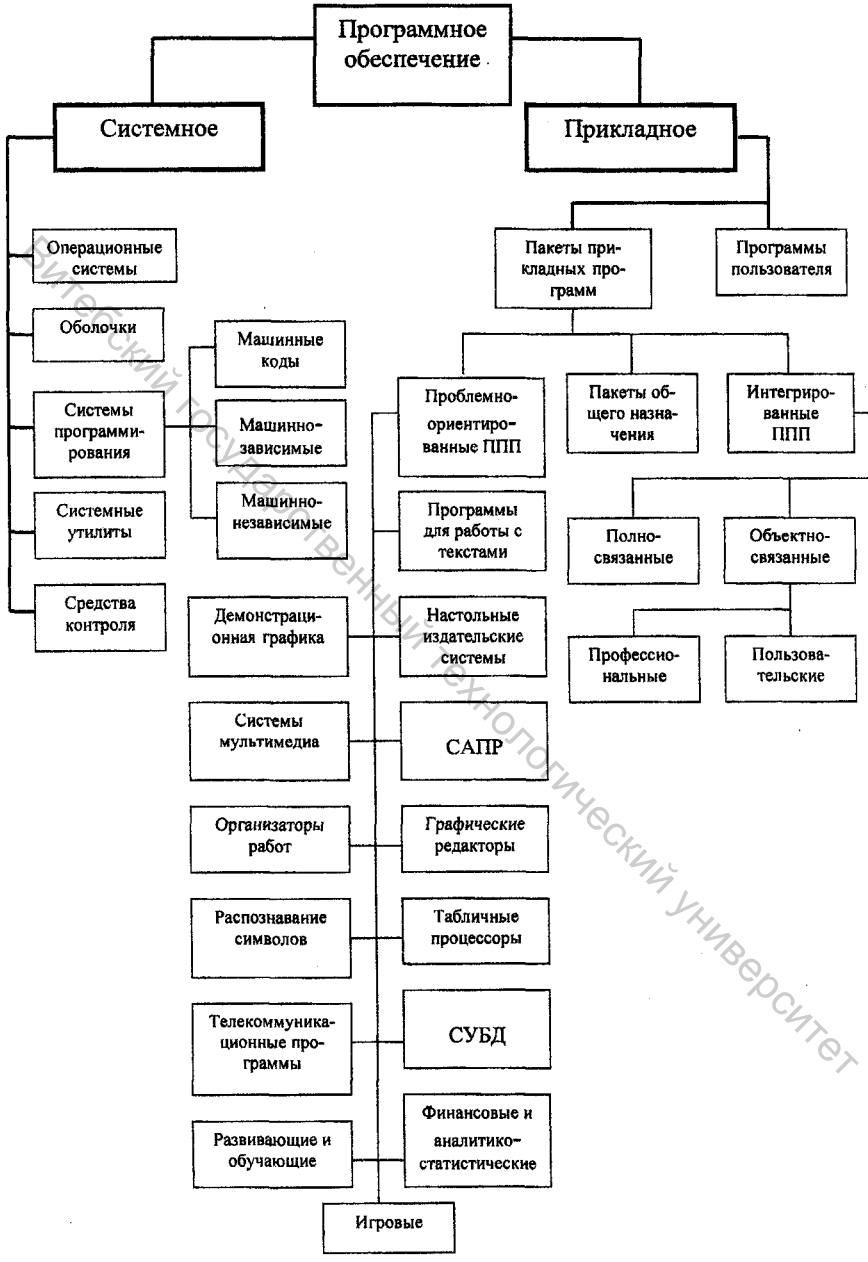


Рис. 3.2. Структура программного обеспечения персонального компьютера

ТЕМА 2

ОСНОВЫ АЛГОРИТМИЗАЦИИ ИНЖЕНЕРНО-ТЕХНИЧЕСКИХ ЗАДАЧ

ГЛАВА 4. ТИПОВЫЕ ПРИЕМЫ АЛГОРИТМИЗАЦИИ

4.1. Алгоритмы и их свойства

Понятие «алгоритм» является одним из основных в информатике. Это слово происходит от имени узбекского математика аль-Хорезми, что означает «из Хорезма». В IX веке он разработал правила арифметических действий над десятичными числами. Длительное время алгоритмами пользовались только математики, понимая под алгоритмом любое описание процесса решения задачи. Описание процессов решения математических задач предназначалось для человека, поэтому не требовалось большой строгости в описании действий.

С развитием техники появились автоматические устройства, способные воспринимать команды и исполнять соответствующие действия. Для таких устройств правила решения должны быть четко сформулированы, то есть исполнителю необходимо указать последовательность действий, которые он должен выполнить для достижения цели – получения результата. Иначе говоря, исполнителю должен быть указан алгоритм решения задачи, представленный на понятном ему языке. Под исполнителем подразумевается как человек, так и вычислительная машина.

Алгоритм решения задачи – это конечная последовательность чётко сформулированных правил решения некоторого класса задач, приводящая к получению результата.

В обыденной жизни мы часто сталкиваемся с алгоритмами как последовательностью действий, приводящих к достижению поставленной цели, – это правила перехода улицы, рецепты приготовления различных блюд, поиск нужного слова в словаре и др. Алгоритмы в математике – это правила нахождения корней квадратных и алгебраических уравнений, правила выполнения арифметических действий, правила нахождения наибольшего общего делителя и наименьшего общего кратного, разложения числа на простые множители и др.

Для адекватности восприятия и понимания исполнителем алгоритмы должны обладать целым рядом свойств: дискретностью, точностью, понятно-

стью, результативностью, массовостью. Таким образом, в качестве основных свойств алгоритма можно выделить следующие:

дискретность – это разбиение алгоритма на ряд отдельных законченных действий – шагов;

точность – это четкое указание последовательности шагов;

понятность – это однозначное понимание и исполнение каждого шага алгоритма его исполнителем;

результативность – обязательное получение результата за конечное число шагов;

массовость – применимость алгоритма к решению целого класса однотипных задач.

Если бы человек знал алгоритм решения всех задач, то их исполнение можно было бы поручить машине. Но оказалось, что не все задачи, которые нам хотелось бы решить, имеют алгоритмы решения. Задачи, в принципе не имеющие общего решения, называют алгоритмически неразрешимыми. Это, например, задача о трисекции угла, о квадратуре круга и др.

4.2. Способы описания алгоритмов

Существуют различные способы представления алгоритмов. Рассмотрим некоторые из них.

Текстовый – алгоритм решения задачи задается в виде текстового описания шагов решения. Таким способом составляются различные инструкции (например, пользование бытовыми приборами, управление автоматическими игрушками, рецепты приготовления блюд и др.). Такие алгоритмы предназначены для человека, поэтому в качестве команд могут использоваться привычные для человека предложения, фразы. Форма записи должна быть такой, чтобы был ясен порядок исполнения команд.

Математический – такой способ описания алгоритма также предназначен для человека и представляет собой логически связанную последовательность математических формул, реализующих решение задачи.

В виде блок-схем – схема алгоритма представляет собой графическое изображение алгоритма на формализованном языке, описывающего содержание и логические связи задачи. Под блоком подразумевается любой конечный этап вычислительного процесса, принимаемый в данной схеме как целое. В блок-схеме все части вычислительного процесса представлены фигурами, содержащими пояснения и соединенными между собой линиями. Применяемые графические символы, отражающие основные операции процесса обработки данных, устанавливает ГОСТ 19.003-80 (обозначение символов соответствует международному стандарту ISO 1028-73). Наиболее часто используемые блоки представлены на рис. 4.1.

Правила выполнения схем алгоритмов регламентирует ГОСТ 19.002-80 (соответствует международному стандарту ISO 2636-73).

Блок-схема должна иметь начало и конец. Связь между блоками отмечается *линиями потока информации*, которые показывают последовательность решения задачи. Основное направление потока информации идет *сверху вниз* и *слева направо*. Здесь стрелки на линиях можно не указывать. Нумерация блоков производится от начала в порядке их расположения на блок-схеме. Направление линий потока, идущих *снизу вверх* и *справа налево*, показывается стрелками. По отношению к блоку линии потока могут быть входящими или выходящими. Количество входящих линий для блока принципиально не ограничено. Выходящая линия может быть только одна. Исключение составляют *логические блоки* (блоки проверки условия), имеющие не менее двух выходящих линий потока, каждая из которых соответствует одному из возможных исходов проверки логического условия. Над линиями выхода обозначается логическое условие выхода: «Да», если условие выполнено, и «Нет», если условие не выполнено, а также блоки модификации.

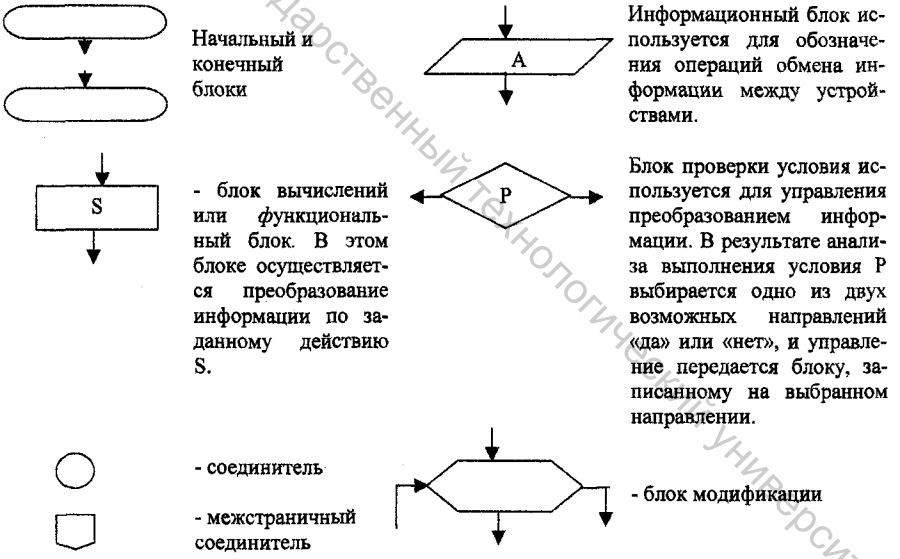


Рис.4.1. Блоки, используемые для составления блок-схем алгоритмов

При значительном количестве пересекающихся линий, большой их длине и многократных изменениях направления для повышения наглядности схемы допускается разрывать линии потока информации, размещая на обоих концах разрыва *соединитель*. Внутри каждого поля соединителя ставится одинаковая маркировка отдельной буквой или буквенно-цифровой координатой блока, к которому подходит линия потока. Если схема располагается на нескольких листах, переход линий потока с одного листа на другой обозначается с помощью

межстраничного соединителя. При этом на листе с блоком-источником соединитель содержит номер листа и координаты блока-приемника, а на листе с блоком-приемником – номер листа и координаты блока-источника. Внутри блоков и рядом с ними проставляют записи и обозначения так, чтобы их можно было читать слева направо и сверху вниз независимо от направления потока.

Еще одним способом описания алгоритма является описание *на языке программирования*: на основании алгоритма, сформулированного одним из перечисленных способов, программистом составляется программа решения поставленной задачи на специальном языке для записи алгоритмов. Такие языки получили название *алгоритмические языки,* или *языки программирования высокого уровня*.

При изображении алгоритмов при помощи схем используются базовые управляющие структуры: *следование, развилка, повторение* (рис. 4.1, 4.3, 4.4).

Следование – структура, означающая, что действия S1 и S2 должны быть исполнены одно за другим.

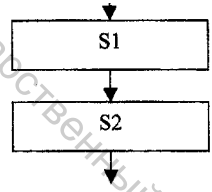


Рис.4.2. Структура следование

Развилка – это действие, определяемое структурой, осуществляющее анализ условия P (истинно или ложно) и альтернативный выбор дальнейшего направления в последовательности выполнения действий в зависимости от значения P. Различают:

полную развилку и *неполную развилку*

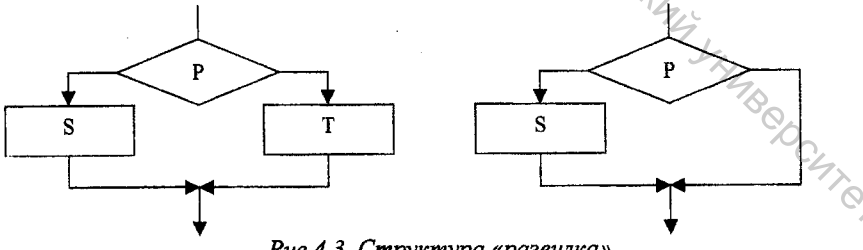


Рис 4.3. Структура «развилка»

Словесно полная развилка описывается так: если условие P истинно, то исполнить S, иначе T.

Неполную развилку словесно можно описать так: если условие P истинно, то исполнять S.

Повторение – это структура, описывающая циклические вычислительные процессы. Различают *цикл-пока* (рис. 4.4а) и *цикл-до* (рис.4.4б).

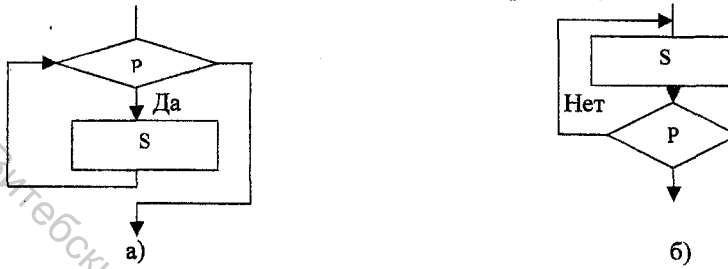


Рис. 4.4. Структура «повторение»: а) цикл-пока, б) цикл-до

Цикл-пока: пока условие P истинно, выполнять тело цикла S.

Цикл-до: выполнять тело цикла S, до тех пор, пока условие P не станет истинным.

Все пять алгоритмических структур могут комбинироваться одна с другой, как того требует алгоритм.

4.3. Этапы подготовки и решения задачи с использованием ПК

Решение любой задачи на ПК представляет собой сложный и трудный процесс, в котором на различных этапах принимают участие различные специалисты: экономисты, инженеры, математики, программисты, статистики и т.д. Этот процесс можно условно разбить на следующие этапы:

1. *Постановка задачи.* На этом этапе специалист формирует цели решения задачи. Подробно раскрывает ее содержание, указывает количество и характер всех величин, используемых в задаче в качестве исходных данных.

2. *Математическая формулировка задачи.* Здесь устанавливаются в окончательном виде те формулы и математические зависимости, которые подлежат решению.

3. *Выбор численного метода.* Среди всех численных методов решения данной задачи выбирается тот из них, который наилучшим образом обеспечивает выполнение всех требований этой задачи.

4. *Разработка логической схемы* алгоритма и ее минимизация. Разрабатывается алгоритм решения, т.е. устанавливается необходимая последовательность арифметических и логических действий, с помощью которых может быть реализован выбранный численный метод. Если возможно, то алгоритм упрощается (минимизируется) с помощью операций и формул математической логики. Полученный алгоритм изображается наглядно в виде блок-схемы.

5. *Программирование.* Алгоритм решения задачи на основании блок-схеме записывается на одном из языков программирования.

6. *Отладка программы.* Задача данного этапа состоит в том, чтобы путем опробования на компьютере вновь разработанной программы выявить ошибки,

допущенные на всех предыдущих этапах. Для проверки правильности вычислений по составленной программе вручную решается один из вариантов задачи, называемый отладочным или контрольным вариантом. Затем этот же вариант по составленной программе просчитывается на ПК. При совпадении результатов счета на ПК и контрольного варианта считают, что программа составлена правильно.

7. *Решение задач на компьютере*. После отладки программы производится непосредственное решение задачи на ПК с целью получения результатов для всех вариантов исходных данных.

8. *Анализ полученных результатов*. Производится специалистом, поставившим задачу. Если результаты по каким-либо причинам не устраивают, принимаются решения либо об изменении алгоритма, либо об уточнении исходных данных.

4.4. Вопросы для самоконтроля

1. Что означает понятие *«алгоритм»?*
2. Перечислите и поясните свойства алгоритмов.
3. Перечислите и поясните способы описания алгоритмов.
4. Какие условные обозначения используются для составления блок-схем?
5. Объясните, в каких случаях используется блок *«соединитель»*, а в каких случаях *«межстраничный соединитель»*.
6. Охарактеризуйте базовую алгоритмическую структуру *«следование»*.
7. Охарактеризуйте базовую алгоритмическую структуру *«развилка»*.
8. Охарактеризуйте базовую алгоритмическую структуру *«повторение»*.
9. Какие блоки используются при изображении структуры *«развилка»*? Охарактеризуйте их.
10. В каком порядке нумеруются блоки на блок-схеме при использовании структуры «развилка»?
11. Какое действие выполняется, если в неполной развилке условие P ложно?
12. Какие действия будут выполняться, если в полной развилке условие P истинно, а действие S представляет собой цикл-пока?
13. Может ли структура *«повторение»* быть вложена в структуру *«развилка»*? Приведите примеры вложенных структур.
14. Сколько раз повторится выполнение тела цикла S в цикле-до, если условие истинно?
15. Какое действие выполнится, если условие P в цикле-пока ложно?
16. Какое действие выполнится, если условие P в цикле-до ложно?
17. Какие блоки используются при изображении структуры *«повторение»*? Охарактеризуйте их.
18. Перечислите этапы подготовки к решению задачи на ПК.
19. Что представляет собой этап отладки программы?
20. Какие виды ошибок могут быть выявлены при отладке программы?

ТЕМА 3

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PASCAL

ГЛАВА 5. ОСНОВНЫЕ ПОНЯТИЯ ЯЗЫКА PASCAL

На протяжении уже многих лет среди программистов достаточно популярен язык программирования Паскаль (Pascal). Он был разработан в 1968-1971 г.г. Никлаусом Виртом первоначально для целей обучения программированию вообще, но благодаря ряду особенностей получил достаточно широкое распространение и в практической работе. В 1973 году был разработан первый транслятор с языка Паскаль. Сегодня ведущие разработчики программных средств Microsoft и Borland регулярно поставляют на рынок новые версии этого языка.

По своей идеологии Паскаль весьма полно отражает идеи структурного программирования. Он хорошо приспособлен для применения технологии разработки программ методом нисходящего проектирования (пошаговой детализации) и предоставляет весьма гибкие возможности в отношении используемых структур данных.

Хотя Паскаль создавался для целей обучения, он хорошо продуман с точки зрения эффективности реализации самого языка и получаемых в результате трансляции машинных программ. Большое внимание в языке уделено вопросу повышения надежности программ: средства языка позволяют осуществлять достаточно полный контроль правильности использования данных различных типов и программных объектов.

Как любой алгоритмический язык, Паскаль образуют три его составляющие: алфавит, синтаксис и семантика [3].

Алфавит – это фиксированный для данного языка набор основных символов, т.е. «букв алфавита», из которых должен состоять любой текст на этом языке, никакие другие символы в тексте не допускаются.

Синтаксис – это система правил, определяющих допустимые конструкции из букв алфавита. С помощью этих конструкций представляются отдельные компоненты алгоритма и алгоритм в целом, записанные на данном языке.

Семантика – это система правил истолкования отдельных языковых конструкций, позволяющих однозначно воспроизвести процесс обработки данных по заданной программе.

Система программирования Turbo Pascal представляет собой интегрированную среду, включающую такие компоненты, как текстовый редактор, ком-

пилятор (транслятор), редактор связей и отладчик. Текст программы, первоначально набранный в редакторе, называется *исходным модулем* (исходным кодом). Исходный код подвергается компиляции – переводится на язык машинных кодов. В результате получается *объектный модуль* (объектный код) – текст программы на машинном языке. Далее редактор связей объединяет отдельные, независимо откомпилированные объектные модули в общую программу. Такая программа называется *загрузочным модулем*. При необходимости на этапах компиляции и компоновки к загрузочному модулю может подключаться отладчик, который позволяет пользователю проконтролировать ход выполнения его программы и выявить имеющиеся ошибки в алгоритме.

5.1. Лексемы языка

Лексема – это минимальная конструкция языка, состоящая из символов алфавита, имеющая смысловое значение. Рассмотрим следующие виды лексем.

Слова. Слова подразделяются на *зарезервированные слова, стандартные идентификаторы и идентификаторы пользователя* (идентифицировать - отождествлять).

Зарезервированные слова являются составной частью языка, имеют фиксированное начертание и раз и навсегда определенный смысл. Всего зарезервированных слов около шестидесяти (*например*, `program, begin, end`).

Стандартные идентификаторы служат для обозначения заранее определенных разработчиками языка типов данных, констант, процедур и функций. (`Sin, Ln`).

Идентификаторы пользователя применяются для обозначения меток, констант, переменных, процедур и функций, определенных самим программистом.

Существуют общие правила написания идентификаторов [1]:

- идентификатор начинается только с буквы или знака подчеркивания;
- идентификатор может состоять из букв, цифр и знака подчеркивания (пробелы, точки и другие специальные символы недопустимы);
- между двумя идентификаторами должен быть хотя бы один пробел;
- максимальная длина идентификатора 127 символов, все они значимы.

При написании идентификаторов можно применять как прописные, так и строчные буквы.

Пример

`SUM2`

`1KL` – ошибка, идентификатор должен начинаться с цифры.

`BL_6`

`NOMER.DOMA` – ошибка, идентификатор содержит точку.

При записи алгоритма решения задачи на языке программирования необходимо знать правила написания и использования элементарных информационных и языковых единиц.

Программа на языке Pascal формируется с помощью конечного набора знаков, образующих алфавит языка. Алфавит языка состоит из прописных и строчных букв латинского алфавита от A до Z и от a до z, знака подчеркивания, десятичных (от 0 до 9) и шестнадцатеричных цифр, специальных символов. Шестнадцатеричные цифры строятся из десятичных цифр и букв от A до F (или от a до f). При написании программ применяются следующие специальные символы:

+ плюс	> больше	{ } фигурные скобки	‘ апостроф
- минус	< меньше	. точка	# номер
* звездочка	[ ] квадратные скобки	, запятая	$ знак денежной единицы
/ дробная черта	( ) круглые скобки	: двоеточие	^ тильда
= равно	@ коммерческое а	; точка с запятой	пробел (обозначения нет)

Комбинации специальных символов могут образовывать *составные символы*:

:= присваивание	<= меньше или равно
<> не равно	>= больше или равно
.. диапазон значений	(. .) альтернатива [ ]
(* *) альтернатива { }	

К *знакам операций* относятся: * (умножение), / (деление), + (сложение), - (вычитание), = (равно), < > (не равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно).

Неделимые последовательности знаков алфавита образуют *слова*, отделенные друг от друга разделителями и несущие определенный смысл в программе.

Разделители используются для того, чтобы придать тексту программы наглядность. Группу разделителей образуют символы: « ;» « ,» « :» « .» .

Для *комментариев*, то есть пояснений к программе, используются символы { } и (* *) – альтернатива фигурных скобок.

5.2. Константы

Константы – это элементы данных, значения которых известны заранее и в процессе выполнения программы не изменяются. Различают такие виды констант, как константы-литералы, именованные и типизированные константы.

К *константам-литералам* относятся:

- десятичные числа (*например*, 69, -619, 21.14 и т.п);
- целые шестнадцатеричные числа, которые начинаются со знака $. *Например*, $B, $3A и т.п.;
- символьные константы ‘A’, ‘45’, ‘?’ и т.п.;
- строковые константы ‘FFF’, ‘DOM’, ‘BOX’ и т.п.

Именованная константа – это идентификатор, которому должно быть присвоено значение. Все именованные константы должны быть описаны в разделе описания констант программы. В языке Pascal для определения констант служит зарезервированное слово `Const`.

Формат

```
Const <идентификатор> = <значение константы>;
```

Пример

```
Const Max = 1000;Arg = -54.69; C = 'GRUPPA';
```

В таблице 5.1 представлен ряд констант, к значениям которых можно обращаться без предварительного определения.

Типизированная константа – это константа, для которой указывается не только значение, но и тип.

Пример

```
Const GR: INTEGER = 2005;
```

где `INTEGER` – целочисленный тип. Типизированной константе можно присваивать только те значения, которые определяются данным типом.

Таблица 5.1. Зарезервированные константы

Идентификатор	Тип	Значение	Описание
`Pi`	`Real` (вещественный)	3,141592..	Число "пи"
`True`	`Boolean` (булевский)	True	"Истина"
`False`	`Boolean` (булевский)	False	"Ложь"
`Maxint`	`Integer` (целочисленный)	32767	Максимальное целое число

5.3. Переменные

Переменная – это величина, которая, в отличие от константы, может менять свое значение в процессе выполнения программы.

Каждая переменная характеризуется:

- *именем переменной*, которое является идентификатором;
- *типом переменной*. *Тип* – это набор значений, которые могут принимать объекты программы, и совокупность операций, допустимых над этими значениями.

Тип констант автоматически распознается компилятором Паскаля без предварительного описания. Тип переменных должен быть описан перед тем, как с переменными будут выполняться какие-либо действия. Для описания переменных предназначено зарезервированное слово `Var`.

Формат

```
Var <идентификатор>: <тип>;
```

Пример

```
Var  C: integer;
```

```
J:Char;
A, B:Real;
```

5.4. Типы данных

Понятие типа является одним из фундаментальных понятий любого языка программирования. Объекты (константы, переменные, функции, выражения), которыми оперирует программа, относятся к определенному типу.

Тип – это множество значений, которые могут принимать объекты программы, и совокупность операций, допустимых над этими значениями [1]. Например, значения 1 и 2 относятся к целочисленному типу, их можно складывать, умножать и выполнять другие арифметические операции. Значения «монитор» и «Паскаль» носят лингвистический характер. Они имеют свой набор допустимых операций. В большинстве широкоупотребительных языков могут использоваться только строго определенные, заранее известные типы. Pascal, наряду со стандартными типами, имеющимися в других языках высокого уровня, позволяет программисту образовывать собственные типы.

Все допустимые в языке Паскаль типы подразделяются на две большие группы: *скалярные* и *структурированные*.

Скалярные типы подразделяются на *стандартные* и *определенные пользователем*.

Стандартные типы предлагаются пользователям разработчиками языка Pascal. К стандартным скалярным типам относятся *целочисленные*, *вещественные*, *литерные*, *булевские* типы данных и *указатели*.

Целочисленные типы данных представляют собой значения, которые могут использоваться в арифметических выражениях и занимают в памяти от 1 до 4 байт. Характеристика целочисленных типов данных представлена в таблице 5.2.

Таблица 5.2. Характеристика целочисленных типов данных

Тип	Диапазон	Требуемая память (байт)
BYTE	0..255	1
SHORTINT	-128..127	1
INTEGER	-32768..32767	2
WORD	0..65535	2
LONGINT	-2147483648..2147483647	4

Данные целочисленных типов могут быть представлены как в десятичной, так и в шестнадцатеричной системах. Если число представлено в шестнадцатеричной системе, перед ним без пробела записывается знак $.

Пример

```
Var  Sort, Nomer: integer;
     A, B, C: integer;
     Gran: byte; Min, Max: byte;
```

Переменная типа Byte занимает в памяти 1 байт, т.е. по сравнению с типом Integer память экономится в два раза. Использование байтового типа целесообразно, если известно, что значение переменной не превысит 255. В арифметических и логических выражениях допустимо смешение типов byte и integer.

Вещественные типы данных представляют собой вещественные значения, которые используются в арифметических выражениях и занимают в памяти от 4 до 6 байт. Паскаль допускает представление вещественных значений с плавающей или фиксированной точкой. Вещественные десятичные числа с *фиксированной точкой* записываются по обычным правилам арифметики. Целая часть отделяется от дробной десятичной *точкой*. Если десятичная точка отсутствует, число считается целым. Вещественные десятичные числа с *плавающей точкой* представлены в экспоненциальном виде: mE+p, где m – мантисса (целое или дробное число с десятичной точкой), E обозначает «10 в степени», p –порядок (целое число).

Характеристика вещественных типов данных представлена в таблице 5.3

Эффективное использование типов SINGLE, DOUBLE, EXTEND, COMP возможно только при включенной директиве {$N+}. По умолчанию она находится в выключенном состоянии. Для решения инженерно-экономических задач достаточно значений типа REAL.

Пример

```
Var Res, Summa, Itog: real;
    X, Y: real;
```

Таблица 5.3. Характеристика вещественных типов данных

Тип	Диапазон	Мантисса, знаков	Требуемая память
REAL	2.9*10E-39..1.7*10E38	11-12	6
SINGLE	1.5*10E-45..3.4*10E38	7-8	4
DOUBLE	5.0*10E-324..1.7*10E308	15-16	8
EXTEND	1.9*10E-4951..1.1*10E4932	19-20	10
COMP	-2E+63+1..2E+63-1	10-20	8

Булевский тип данных описывается стандартным идентификатором BOOLEAN. Переменные и константы этого типа могут принимать только одно из двух значений: TRUE (истина) или FALSE (ложь).

Пример

```
Var Sel1, Sel2: boolean;
A,B,C,D: boolean;
```

Выражения булевского типа занимают в памяти 1 байт и используются в логических выражениях и выражениях отношения, а также для управления порядком выполнения операторов программы.

Литерный (символьный) тип описывается стандартным идентификатором CHAR. Константы и переменные этого типа могут принимать одно из значений кодовой таблицы ASCII, символ берется в апострофы.

Пример

```
Var Bukva, Znak, Simvol: char; Ch: char;
```

Переменные символьного типа занимают в памяти 1 байт. Использование данных типа char в арифметических выражениях запрещено. К литерным значениям могут применяться операции сравнения, результат при этом зависит от номера литерной переменной или константы в кодовой таблице.

Кроме стандартных типов данных, Паскаль поддерживает скалярные типы, *определенные пользователем*.

К ним относятся *перечисляемый* и *интервальный* типы. Данные этих типов занимают в памяти 1 байт, поэтому любой пользовательский тип не может содержать более 255 элементов. Их применение значительно улучшает наглядность программы, делает более легким поиск ошибок и экономит память.

Перечисляемый тип задается непосредственно перечислением всех значений, которые может принимать переменная данного типа. Отдельные значения указываются через запятую, а весь список заключается в круглые скобки.

Формат

```
Type <имя типа>=(<значение1, значение2, ..., значениеn>);
Var <идентификатор, ...>: <имя типа>;
```

Пример

```
Type Gaz = (C, O, N, F);
     Metall =(Fe, Co, Na, Cu, Zn);
Var G1, G2, G3  : Gaz;
    Met1, Met2 :  Metall;
    Season: (Winter, Spring, Summer, Autumn);
```

В данном примере [3] приведены два явно описанных типа данных пользователя – Gaz и Metall. Определены их значения – обозначения некоторых газов и металлов периодической таблицы Менделеева. Переменные G1, G2, G3, Met1, Met2 могут принимать только одно из перечисленных значений. Попытка присвоить им любое другое значение вызовет программное прерывание. Третий тип перечисления – анонимный (не имеет имени) и задается перечислением значений в разделе Var. Season является переменной этого типа и может принимать значения Winter, Spring, Summer, Autumn. Таким образом, может быть задан любой тип, но это не всегда приемлемо. Первый способ, безусловно, более понятен и больше соответствует характеру языка Pascal.

Интервальный тип позволяет задавать две константы, определяющие границы диапазона значений для данной переменной. Компилятор при каждой операции с переменной интервального типа генерирует подпрограммы проверки, определяющие, остается ли значение переменной внутри установленного для нее диапазона. Обе константы должны принадлежать одному из стандарт-

ных типов, кроме вещественного. Значение первой константы должно быть обязательно меньше значения второй.

Формат

```
Type <имя типа> = <константа1> .. <константа2>;
     Var <идентификатор>: < имя типа>;
```

Пример

```
Type Days = 1.. 31;
  Var Work_d, Free_d: Days;
```

В этом примере переменные *Work_d, Free_d* имеют тип *Days* и могут принимать любые значения из диапазона 1 . . 31. Выход из диапазона вызывает программное прерывание.

Можно определить интервальный тип, задав границы диапазона не значениями констант, а их именами:

```
Const Min = 1; Max = 31;
Type Days = Min .. Max;
  Var Work_d, Free_d: Days;
```

Структурированные типы данных базируются на скалярных типах и могут содержать их различные комбинации. Они определяют упорядоченную совокупность скалярных элементов и характеризуются типом своих компонентов [1]. В языке Pascal представлены следующие структурированные типы данных:

строка – последовательность символов, заключенная в апострофы;

массив – структурированный тип данных, состоящий из фиксированного количества элементов одного и того же типа, доступ к которым осуществляется по индексу;

множество – набор выбранных по какому-либо признаку или группе признаков объектов, которые можно рассматривать как единое целое;

запись – совокупность фиксированного числа компонентов разного типа;

указатель – неограниченное множество, указывающее на однотипные элементы значений;

файл – последовательность компонентов одного типа и одной длины.

5.5. Вопросы для самоконтроля

1. Дайте определение понятия «лексема» языка Pascal? Охарактеризуйте виды лексем.
2. Как правильно записать идентификатор на языке Pascal?
3. Какие символы составляют алфавит языка Pascal?
4. Какие знаки операций могут использоваться при составлении программ на языке Pascal?
5. Дайте определение понятия «константа» языка Pascal. Охарактеризуйте виды констант.
6. Все ли константы языка Pascal должны быть предварительно описаны в разделе описаний программы?

7. Дайте определение понятия «переменная» языка Pascal.
8. Приведите форматы описания констант и переменных в языке Pascal.
9. Дайте определение понятия «тип данных». На какие группы подразделяются все допустимые в языке Pascal типы данных?
10. Данные каких типов относятся к стандартным скалярным типам?
11. Приведите форматы описания целочисленного и байтового типов данных. В чем состоит различие этих типов данных?
12. Приведите формат описания вещественного типа данных. Охарактеризуйте этот тип данных.
13. Приведите формат описания логического типа данных. Охарактеризуйте этот тип данных.
14. Приведите формат описания литерного (символьного) типа данных. Охарактеризуйте этот тип данных.
15. Данные каких типов относятся к скалярным типам, определенным пользователем?
16. Приведите формат описания литерного (символьного) типа данных. Охарактеризуйте этот тип данных.
17. Приведите формат описания перечисляемого типа данных. Охарактеризуйте этот тип данных.
18. Приведите формат описания интервального типа данных. Охарактеризуйте этот тип данных.
19. Поясните на примере различие в описании и использовании перечисляемого и интервального типов данных.
20. Перечислите и охарактеризуйте структурированные типы данных языка Pascal.

ГЛАВА 6. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

6.1. Арифметические выражения и правила их записи

Выражение задает порядок выполнения действий над элементами данных и состоит из операндов, круглых скобок и знаков операций. В качестве операндов выступают константы, переменные и обращения к функциям. *Арифметическое* выражение возвращает целое или действительное значение и представляет собой набор операндов, соединенных знаками арифметических операций [4].

Арифметические операции выполняют арифметические действия в выражениях над операндами числовых типов. Основные арифметические операции языка Pascal представлены в таблице 6.1.

Таблица 6.1. Арифметические операции языка Pascal

*	умножение	
/	деление	
+	сложение	
-	вычитание	
DIV	операция целочисленного деления	Эти операции используются только с данными целочисленного типа
MOD	остаток от целочисленного деления	
AND	арифметическое И	
OR	арифметическое ИЛИ	

Операции DIV и MOD выполняются следующим образом:

36 div 5 = 7 - выделяет целую часть полученного частного;

36 mod 5 = 1 - выделяет остаток от деления.

В арифметических выражениях могут использоваться стандартные функции, поддерживаемые языком Pascal (таблица 6.2).

Таблица 6.2. Стандартные функции языка Pascal

Математическая запись	Запись на Турбо Паскале	Примечание
sin x	Sin(X)	
cos x	Cos(X)	
Ln x	Ln(X)	x>0
$\sqrt{x}$	Sqrt(X)	$x \geq 0$
e^x	Exp(X)	
x^2	Sqr(X)	
arctg x	Arctan(X)	$-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$
$\lvert x \rvert$	Abs(X)	$x = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$

Аргумент X может быть любого числового типа. При решении некоторых задач могут быть необходимы следующие функции [3]:

- получение целой части X:

  `Trunc(X)` – результат имеет целочисленный тип;

  `Int(X)` – результат имеет вещественный тип;
- получение дробной части X: `Frac(X)` – результат вещественного типа;
- округление X до целого: `Round(X)` – результат имеет вещественный тип;
- получение значения случайного числа

  `Random` – генерирует случайное значение из диапазона 0..0,99, результат имеет вещественный тип;

  `Random(X)` – генерирует случайное значение из диапазона от 0 до x, результат имеет целочисленный тип.

  Для обновления базы генерации используется процедура `Randomize`.
- `Pred(X)` – возвращает элемент, предшествующий X в списке значений типа. Тип результата совпадает с типом параметра;
- `Succ(X)` – возвращает значение, следующее за X в списке значений типа. Тип результата совпадает с типом параметра.

Особый интерес представляет функция `ODD`, служащая для определения четности числа. Ее аргументом может быть переменная только целого типа.

`ODD(X)` – возвращает значение булевского типа, равное `True`, если X – нечетное и `False`, если X – четное.

Пример

Выражение	Результат
`ODD(3)`	`True`
`ODD(4)`	`False`
`ODD(2+7-1)`	`False`

Если в выражениях имеются нестандартные функции, они должны быть выражены через стандартные следующим образом:

$x^k \Rightarrow e^{k \cdot \ln x} \Rightarrow \exp(k * \ln(x))$;

$a^x \Rightarrow e^{x \cdot \ln a} \Rightarrow \exp(x * \ln(a))$;

$\log_a x \Rightarrow \ln x / \ln a \Rightarrow \ln(x)/\ln(a)$;

$\lg x \Rightarrow \ln x / \ln 10 \Rightarrow \ln(x)/\ln(10)$;

$tgx \Rightarrow \sin x / \cos x \Rightarrow \sin(x)/\cos(x)$;

Например,

$$\sqrt[5]{ax+2} \Rightarrow (ax+2)^{\frac{1}{5}} \Rightarrow \frac{1}{5} * \ln(ax+2) \Rightarrow e^{\frac{1}{5} * \ln(ax+2)} \Rightarrow \exp(\frac{1}{5} * \ln(ax+2))$$

$e^{\sqrt{2x+4}} = \exp(sqrt(2x+4))$

6.2. Операторы языка Turbo Pascal

Основная часть программы на языке Паскаль представляет собой последовательность операторов.

Оператор – это конструкция языка, которая определяет действия, выполняемые над данными, или порядок выполнения этих действий [7]. Операторы выполняются последовательно в том порядке, в котором они записаны в тексте программы. Разделителем операторов служит точка с запятой. Все операторы языка Турбо Паскаль разделяются на три группы:

1. *Простые операторы.*
2. *Операторы ввода-вывода*, которые на самом деле являются процедурами, так как в них можно указывать параметры.
3. *Структурированные* (сложные) операторы, в состав которых включаются другие операторы по строго определенным правилам.

Простые операторы – это операторы, не содержащие в себе никаких других операторов. К ним относятся операторы *присваивания, безусловного перехода, вызова процедуры и пустой оператор.*

Оператор присваивания (:=) предписывает выполнить выражение, заданное в его правой части, и присвоить результат переменной, идентификатор которой расположен в левой части. Переменная и выражение должны иметь один и тот же тип. Исключение представляет случай, когда переменная имеет вещественный тип, а выражение – целочисленный. Допустимо присваивание любых типов данных, кроме файловых.

Формат

```
<идентификатор>:=<выражение>;
```

Пример

```
Sort:=1;
Cena:=15.23;
Nazv:='Model N986';
Res:=Sin(A)+Cos(B);
```

Оператор безусловного перехода GOTO означает «перейти к» и применяется в случаях, когда после выполнения некоторого оператора надо выполнить не следующий по порядку, а какой-нибудь другой, отмеченный меткой оператор. Метка может содержать как цифровые, так и буквенные символы [1].

Формат

```
GOTO <метка>;
```

Пример

```
...
Label Metka1, Metka2;
...
Metka1: GOTO Metka2;
```

```
Metka2: GOTO Metka1;
...
```

При записи оператора GOTO необходимо помнить следующее.

Метка, на которую передается управление, должна быть описана в разделе описания меток того блока процедуры, функции, основной программы, в котором эта метка используется. Областью действия метки является тот блок, в котором она описана. Попытка выйти за пределы блока вызывает программное прерывание.

Обычно оператор GOTO применяется для преждевременного выхода из цикла или обработке ошибок. Но частое использование оператора GOTO усложняет понимание логики программы.

Оператор вызова процедуры. Служит для активизации предварительно определенной пользователем или стандартной процедуры.

Формат

```
<имя процедуры> {(список параметров)};
```

Пример

```
Program Prim;
Procedure V1;
Begin
. . .                          {тело процедуры V1}
end;
Procedure V2;
Begin
. . .                          {тело процедуры V2}
end;
Begin
V1;                    {вызов для выполнения процедуры V1}
V2;                    {вызов для выполнения процедуры V1}
End.
```

Пустой оператор не содержит никаких символов и не выполняет никаких действий. Он может быть расположен в любом месте программы, где синтаксис языка допускает наличие оператора. Как и все другие операторы, пустой оператор может быть помечен меткой. Чаще всего пустой оператор используется для организации выхода из середины программы или составного оператора.

Пример

```
Begin
  Goto Metka;          {переход в конец блока}
. . .
   Metka;                    {пустой оператор помечен меткой}
End.
```

Операторы ввода-вывода. Решение самой простой задачи не обходится без операций ввода-вывода информации. *Ввод данных* – это передача информации от внешнего носителя в оперативную память для обработки. *Вывод* – обратный процесс, когда данные передаются после обработки из оперативной памяти на внешний носитель.

Для выполнения операций ввода-вывода служат четыре оператора: `Read, Readln, Write, Writeln` [6].

Для ввода данных с клавиатуры используется оператор чтения `Read`.

Формат

```
Read(список ввода);
```

где в списке ввода могут фигурировать только переменные допустимых типов данных.

Пример

```
Var I: integer;
    J: integer;
    K: char;
Begin
   read(I, J, K);
```

Оператор чтения `Readln` аналогичен оператору `Read`. Отличие состоит в том, что после считывания последнего в списке значения для одного оператора `Readln` данные для следующего оператора `Readln` будут считываться с начала новой строки.

Оператор записи `Write` производит вывод числовых данных, символов, строк, булевских значений.

Формат

```
Write(список вывода);
```

где список вывода представляет собой набор переменных и выражений.

Например,

`Write(y1, y2, sin(3), yn)` – выражения типа integer, byte, real, char, boolean и т.д.

Оператор вывода `Write` может использоваться в разных *форматах*.

Для вывода переменных *целого* и *символьного* типов используется следующий формат: `Write(X:P)`, где параметр P определяет, сколько знакомест на экране будет выделено для вывода значения той или иной переменной. *Например*, `Write(X:3,C:7)`

Для переменных *вещественного* типа используется обычно следующий формат: `Write(X:P:q)`, где P означает общее число позиций на экране, выделенное для данной переменной, а q – число позиций (из общего числа) после десятичной точки. *Например*, `Write(X:3:1, C:7:4)` – правильный формат, `Write(X:1:3, C:7:6)` – неправильный формат.

Для того, чтобы хорошо и понятно оформить вывод результирующей информации на экран, иногда требуется вывести текстовую информацию, поясняющую результат.

Пример

```
Write('Сумма ряда =', S:7:3, 'при X=',X:4:2);
```

Оператор `Writeln` аналогичен оператору `Write`, но после вывода последнего в списке значения происходит перевод курсора к началу следующей строки. Оператор `Writeln` без параметров вызывает пропуск пустой строки.

Знание рассмотренных операторов дает возможность программирования алгоритмов линейной структуры. Но для того, чтобы правильно составить программу на языке Pascal, необходимо знать общие правила написания программ, соблюдение которых обязательно.

6.3. Структура программы на языке Pascal

Любая программа реализует алгоритм решения задачи. Она объединяет последовательность действий, выполняемых над определенными данными с помощью определенных операций для реализации определенной цели. Основные характеристики программы: точность полученного результата, время выполнения и объем требуемой памяти.

Синтаксически программа на языке Pascal состоит из двух частей: *заголовка*, который не является обязательным, и *блока*, который может содержать в себе другие блоки.

Блок, в свою очередь, состоит из двух частей: *раздела описаний* и *раздела операторов*. Первая часть может отсутствовать, без второй блок не имеет смысла. Блок, который не входит ни в какой другой блок, называется *глобальным*. Если в глобальном блоке находятся другие блоки, они называются *локальными*.

Глобальный блок – это основная программа, он должен присутствовать в любом случае.

Локальные блоки – это процедуры и функции, их присутствие необязательно.

В начале программы находится *заголовок*, который в общем случае состоит из зарезервированного слова `Program`, имени программы и параметров, с помощью которых программа взаимодействует с операционной системой. Заголовок программы несет чисто смысловую нагрузку и может отсутствовать.

Формат заголовка программы

```
Program <имя программы>;
```

где имя программы является идентификатором.

Например, `Program Lin;`

После заголовка следует *программный блок*, состоящий в общем случае из шести разделов: описания меток, описания констант, определения типов данных, описания переменных, описания процедур и функций, раздела операторов.

Любой раздел, кроме раздела операторов, может отсутствовать. Разделы описаний могут встречаться в программе любое количество раз и следовать в любом порядке. Главное, чтобы все описания объектов программы были сделаны до того, как они будут использованы.

Раздел описания меток

Перед любым оператором языка Pascal можно поставить метку, что позволяет выполнить прямой переход на этот оператор с помощью оператора перехода `goto` из любого места программы. Метка состоит из имени и следующего за ним двоеточия. Именем может служить идентификатор или цифра.

Перед употреблением метка должна быть описана.

Формат

```
Label <имя, . . . >;
```

Пример

```
Label M1, M2, 4, Blok2;
```

Далее в тексте программы выполняется обращение к меткам, для этого обычно используется оператор `Goto`.

Например,

```
Goto 20
. . .
20:  Y:=2*SIN(X);
```

Здесь 20 – это метка оператора.

Раздел описания констант

В этом разделе производится присваивание идентификаторам констант постоянных значений.

Формат

```
Const < идентификатор > = < значение>;
```

Пример

```
Const R1=22;
      St=3.25;
      R2:Integer=4;
```

После того как константа определена, ей нельзя присвоить какое-либо другое значение. Ряд констант (`True, False, Maxint`) определен стандартно, к ним можно обращаться без предварительного описания.

Раздел описания типов данных

Этот раздел включается в программу тогда, когда программист использует в программе свои собственные типы. Стандартные типы не требуют описания, в отличие от типов, образованных пользователем. Синтаксис языка Pascal не требует обязательного определения идентификатора типа, так как тип всегда можно задать перечислением в разделе описания переменных.

Формат

```
Type < имя типа > = < значения типа >;
```

Пример

```
Type T1=1..25;
     T2=Array[1..10] of real;
```

Раздел описания переменных

Каждая встречающаяся в программе переменная должна быть описана. Раздел описания переменных включает в себя описание всех переменных, встречаемых в программе.

Формат

```
Var <идентификатор, . . .>: < тип >;
```

Пример

```
Var A,B,X : real;
    Z,Y : integer;
```

Слова `Label,Const, Type` и `Var` являются зарезервированными и обязательно должны присутствовать в начале соответствующих разделов описаний.

Раздел описания процедур и функций

В разделе описания процедур и функций должны быть приведены тексты тех процедур и функций, которые разработаны самим программистом и используются в данной программе.

Стандартные процедуры и функции в программе не описываются, но следует знать, в каких библиотечных модулях они находятся. Имя этого модуля должно указываться сразу за заголовком программы в разделе `Uses`.

Пример

```
Program Lin;
    Uses Crt, Printer;
```

Здесь в модуле `Crt` содержатся стандартные процедуры управления экраном, а модуль `Printer` служит для управления работой принтера.

Характерной чертой языка Pascal является наличие большого количества стандартных процедур и функций (общее название – подпрограммы). Примером стандартных функций являются `Sin(x), Cos(x), Ln(x)` и т.д. Все они находятся в библиотечном модуле `System`. При обращении к функциям аргумент указывается в скобках.

Рассмотрим несколько процедур, которые затем будут использоваться при написании программ:

1. `CLRSCR` – очистка экрана. Курсор перемещается в левый верхний угол экрана.
2. `GOTOXY(N1, N2)` – перемещение курсора в заданную точку экрана. Здесь `N1`, `N2` – параметры:
   - N1 – номер столбца на экране дисплея, целое, лежит в диапазоне от 1 до 80;
   - N2 – номер строки на экране дисплея, целое, лежит в диапазоне от 1 до 25.

Заметим, что левый верхний угол экрана имеет координаты (1,1), а правый нижний угол экрана – (80,25). *Например,* `GOTOXY(20,10)` – перемещение курсора в 10 строку и 20 столбец.

3. `DELAY(T)` – задержка выполнения программы на T микросекунд. *Например*, чтобы сделать задержку на 2 сек, следует записать `DELAY (2000)`.

Все рассмотренные процедуры находятся в библиотечном модуле CRT. В общем случае, для того, чтобы использовать ту или иную процедуру или функцию, необходимо знать:

- имя процедуры (функции);
- количество, порядок следования и назначения параметров процедуры (функции). Значения параметров записываются через запятую;
- в каком библиотечном модуле находится используемая процедура (функция).

Раздел операторов

В программе на языке Pascal раздел операторов является основным, так как именно в нем с предварительно описанными переменными, константами, значениями функций выполняются действия, позволяющие получить результат, ради которого создавалась программа. Здесь описывается алгоритм решения задачи. Этот раздел начинается со слова `Begin` и заканчивается словом `End`, между которыми располагаются строки операторов, описывающих, в каком порядке выполняются действия над данными, чтобы получить требуемый результат. В конце каждого оператора ставится точка с запятой («;»). Завершает раздел зарезервированное слово `End`, после которого стоит точка.

Пример

```
Begin
   < оператор;>
     . . .
   < оператор;>
End.
```

Операторы выполняются строго последовательно в том порядке, в котором они записаны в тексте программы в соответствии с синтаксисом и правилами пунктуации.

В качестве примера составим и разберем простейшую программу на примере линейного алгоритма.

6.4. Пример реализации линейного вычислительного процесса на языке Turbo Pascal

Вычислить значение функции $Y = \sqrt[5]{\sin(x+2)} \cdot \log_3(2 \cdot x) + e^{3x+1}$. Значение X задать самостоятельно.

Вычисление значения Y проведем в несколько этапов. С целью упрощения процесса отладки представим функцию Y в виде трех промежуточных величин: $Y1 = \sqrt[5]{\sin(x+2)}$, $Y2 = \log_3(2 \cdot x)$, $Y3 = e^{3x+1}$, Y=Y1*Y2+Y3.

Текст программы представлен ниже.

```
Program Lin;
 Uses CRT;      (*Раздел подключения стандартных модулей*)
   Var x,y,y1,y2,y3:real;      (*Раздел описания переменных*)
 Begin (*Начало раздела операторов программы*)
(*Ввод исходных данных*)
   Writeln('Введите x');
   Readln(x);
(*Вычисление значения функции Y*)
   y1:= exp(1/5*ln(sin(x+2)));
   y2:=ln(2*x)/ln(3);
   y3:=exp(3*x+1);
   y:=y1*y2+y3;
(*Вывод результатов*)
Writeln('x=',x:4:2,'  y=',y:5:2);
Delay(5000);          (*Задержка результата на экране*)
 End.
```

Результат: *x= 0.55 y=14.23*

6.5. Задания для самостоятельной работы

Составить линейную программу на языке Паскаль для вычисления значений функций a и b.

Вариант	Задание	Вариант	Задание
1	$a=\ln\left(y^{-\sqrt{\lvert x\rvert}}\right)\cdot\left(\sin x+e^{x+y}\right)$ $b=\sqrt{c\left(\sqrt{y}+x^2\right)}\cdot\left(\cos x-\lvert c-y\rvert\right)$	6	$a=0{,}5x^5+3\cos(x+y)+e^{-0{,}1y-x}-\sqrt{\lvert x\cdot y\rvert}$ $b=\sqrt{\left\lvert\frac{-3\cdot tg(x)\cdot\lg\left(x^4+y\right)}{e^{-x}}+1\right\rvert}$
2	$a=arctg(x)-\frac{3}{5}e^{x-y}+0{,}5\frac{\lvert x+y\rvert}{(x+y)^e}$ $b=\frac{e^{\lvert x-y\rvert}\cdot tg(z)}{arctg(y)+\sqrt{x}}+\ln x$	7	$a=\sqrt{e^x+tg(x)+1}\cdot\left(\lg y+\cos(x\cdot y)+\sqrt[3]{x}\right)$ $b=\frac{\lg x-e^{x+y}}{\sqrt{2}+y^2+\lvert x^3-\ln y\rvert}$
3	$a=\frac{(\cos x-\sin y)^3}{\sqrt{tg(z)}}+\ln^2(x\cdot y\cdot z)$ $b=y^x+\sqrt{\lvert x\rvert+e^y}-\frac{z^3\cdot\sin^2 y}{y+\frac{z^2}{y-x}}$	8	$a=\sqrt{12x^4-3x^3+4x^2-5x+6}-\lg^2 z$ $b=\lg\lvert 1-2x+3x^2-4x^3\rvert+\frac{\sqrt{\lvert x\rvert}}{z}$
4	$a=\frac{1+\cos(x+y)}{\left\lvert e^x-\frac{2y}{1+x^2y^2}\right\rvert}\cdot x^3+arctg(y)$ $b=2+\frac{x^2}{\sqrt{2}}+\frac{\lvert y\rvert^3}{\sqrt{2}}+\frac{z^4\cdot(\ln x+1)\cdot\sqrt{2}}{\sqrt{3}}$	9	$a=\frac{2\cos\left(x-\frac{1}{6}\right)}{\frac{1}{2}+\sin^2 y}-\frac{1}{\left\lvert\frac{x^2}{y+x^3}\right\rvert}$ $b=\frac{x\cdot y\cdot z-y\cdot\lvert x+\sqrt{z}\rvert}{10^7+\sqrt[4]{\lg 4}}$
5	$a=\left((1+y)\cdot\sqrt{\sin^2 z+3}-\frac{\lvert y-x\rvert}{5}\right)^3$ $b=\ln\left\lvert\left(y-\sqrt{\lvert x\rvert}\right)\cdot\left(x-\frac{y}{z+\frac{x^2}{4}}\right)\right\rvert$	10	$a=\frac{(x+y-z)^3-(x-y+z)^2+\sqrt{\lvert x+y+z\rvert}}{\log_2(tg(z))}$ $b=\frac{\frac{x}{y}\cdot(z+x)\cdot e^{\lvert x-y\rvert}+\ln(1+x)}{\sin^2 y-(\sin x\cdot\sin y)^2}$

Вариант 11. Составить линейную программу на языке Паскаль, позволяющую определить размеры суровой ткани (длину и ширину),

Ширина суровой ткани, см	Длина суровой ткани, м
$B_c=\frac{B_г\cdot 100}{100-U_y}$	$L_c=\frac{L_т\cdot 100}{100-U_o}$

если известны следующие параметры готовой ткани:

$B_г$ – ширина готовой ткани – 140 см;

U_y – усадка по ширине в отделке – 5%;

$L_т$ – длина куска готовой ткани – 70,2 м;

U_o – усадка по длине в отделке – 3%.

Вариант 12. Составить линейную программу на языке Паскаль, позволяющую определить плотность суровой ткани $P_{ос}$ и Рус, если известны следующие параметры готовой ткани:

$P_{ог}$ – плотность по основе готовой ткани – 11 нит/см;

$P_{уг}$ – плотность по утку готовой ткани – 10 нит/см;

$U_у$ – усадка по утку в отделке – 5%;

$U_о$ – усадка по основе в отделке – 3%.

Плотность по основе, нит/см	Плотность по утку, нит/см
$P_{ос} = P_{ог}\left(1 - \frac{U_о}{100}\right)$	$P_{ус} = P_{уг}\left(1 - \frac{U_у}{100}\right)$

Вариант 13. Составить линейную программу на языке Паскаль, позволяющую определить длину готового куска ткани $L_т$ с учетом уработки основы длиной $L_{ок}$, идущей на выработку одного куска ткани.

Длина готового куска ткани, м	Длина суровой ткани, м	Уработка нитей основы, %
$L_г = L_с\left(1 - \frac{U_о}{100}\right)$	$L_с = L_{ок}\left(1 - \frac{a_о}{100}\right)$	$a_о = \frac{L_о - L_{ос}}{L_о}100$

где $U_о$ – усадка по длине в отделке – 3%;

$L_{ок}$ – длина основы для производства ткани, $L_{ок}$ = 40 м;

$a_о$ – уработка нитей основы, %.

$L_о$ – длина распрямленной нити основы, вынутой из образца ткани, $L_о$=10,5 см;

$L_{ос}$ – длина образца ткани вдоль основы, $L_{ос}$ = 10 см.

Вариант 14. Составить линейную программу на языке Паскаль, позволяющую определить ширину заправки ткани по берду $B_з$, если ширина суровой ткани $B_с$= 145 см, $L_у$ – длина распрямленной нити утка из образца ткани, $L_у$=10,8 см, $L_{ус}$ – длина образца ткани вдоль утка, $L_{ус}$ = 10 см.

Ширина заправки ткани по берду	Уработка нитей по утку, %
$B_з = \frac{B_с}{\left(1 - \frac{a_у}{100}\right)}$	$a_у = \frac{L_у - L_{ус}}{L_у}100$

Вариант 15. Составить линейную программу на языке Паскаль, позволяющую определить длину обычной и ажурной основы $L_{ок}$, необходимой для выработки кусков ткани заданной длины $L_с$.

$L_с$ – длина суровой ткани, $L_с$ = 38 м;

$L_о$ – длина распрямленной нити основы, вынутой из образца ткани, $L_о$=10,5 см;

$L_{ос}$ – длина образца ткани вдоль основы, $L_{ос}$ = 10 см.

Длина обычной основы, м	Уработка нитей основы, %,	Длина ажурной основы, м
$L_{ок} = \dfrac{L_c}{\left(1 - \dfrac{a_o}{100}\right)}$	$a_o = \dfrac{L_o - L_{ос}}{L_o} 100$	$L_{ок} = L_c - \dfrac{a_o}{100}$

Вариант 16. Длина куска подкладочной ткани артикула С21 составляет 41,9 м. Определить количество изделий, которое можно раскроить из данного куска, если норма на обмелку 1 изделия составляет 2,41 м. Определить длину концевого остатка, полученного в результате раскроя полотна. Составить линейную программу на языке Паскаль, позволяющую определить количество кусков указанной длины, необходимое для раскроя 42 изделий. Определить общую длину концевых остатков, образующихся в результате раскроя полученного количества кусков.

Количество изделий = Длина куска / Норма на обмелку.

Полученное значение округлить до ближайшего меньшего целого.

Длина концевого остатка = Длина куска – Количество изделий * Норма на обмелку.

Количество кусков = 42 / количество изделий.

Общая длина концевых остатков = Количество кусков * (Длина куска – Количество изделий * Норма на обмелку).

Вариант 17. Найти расчетное и фактическое количество рабочих 3 разряда, необходимое для выполнения технологически неделимой операции, если такт потока τ составляет 75 с, затрата времени на операцию Т = 117 с. Определить расценку на выполнение данной операции, если стоимость 1 секунды работы 1 разряда равна 3 рубля, а тарифный коэффициент рабочего 3 разряда равен 2,03.

Расчетное количество рабочих $N_p = \dfrac{T}{\tau}$.

Фактическое количество рабочих $N_ф$ принимается округлением расчетного количества рабочих до целого по правилам математики.

Стоимость 1 секунды работы 3 разряда = стоимость 1 секунды 1 разряда * тарифный коэффициент рабочего 3 разряда.

Расценка на выполнение операции = Стоимость 1 секунды * Затрата времени.

Вариант 18.

Расчетное количество рабочих 2 разряда, выполняющих машинные операции, равно 1,55. Расчетное количество рабочих 2 разряда, выполняющих утюжильные операции, равно 0,67. Расчетное количество рабочих 2 разряда, выполняющих ручные операции, равно 1,07. Составить линейную программу на языке Паскаль, позволяющую определить общее количество рабочих 2 разряда, удельный вес каждой специальности в %, сумму разрядов и сумму тариф-

ных коэффициентов, если тарифный коэффициент рабочего 2 разряда равен 1,56.

Общее количество рабочих равно сумме числа рабочих, выполняющих машинные, ручные и утюжильные операции.

Удельный вес каждой специальности = количество рабочих специальности / общее количество рабочих * 100.

Сумма разрядов = Разряд * Общее количество рабочих.

Сумма тарифных коэффициентов = Сумма разрядов * Тарифный коэффициент.

Вариант 19. Составить линейную программу на языке Паскаль, позволяющую определить основные технико-экономические показатели швейного потока: мощность потока в смену, расчетное и фактическое количество рабочих, коэффициент использования рабочего времени и производительность труда 1 рабочего, если такт потока ι составляет 75 с, трудоемкость изготовления изделия Т – 5574 с, время смены R – 28800 с.

Мощность потока (округлить до целого)	Расчетное количество рабочих	Фактическое количество рабочих $N_ф$	Коэффициент использования рабочего времени	Производительность труда 1 рабочего
$M = \frac{R}{\tau}$	$N_p = \frac{T}{\tau}$	округлить N_p до целого	$K_{исп} = \frac{N_p}{N_ф}$	$П_т = \frac{M}{N_ф}$

Вариант 20. Составить линейную программу на языке Паскаль, позволяющую определить экономические показатели проекта строительства предприятия, если коэффициент реновации $К_р$ = 0,0175, нормативный коэффициент экономической эффективности $К_{эн}$ = 0,1.

Капитальные затраты в год $К_{зг} = К_р * (К – Л) + К_{эн} * К$,

где К – единовременные затраты по строительству предприятия, К = 301 тыс. у.е.;

Л – ликвидационное сальдо основных фондов в конце срока службы, Л = 18,7 тыс. у.е.

Капитальные затраты в сутки $К_{зс} = К_{зг} / 365$.

Общая величина условно-постоянных затрат в сутки $З_о = С_о + К_{зс}$,

где $С_о$ – условно-постоянные затраты, $С_о$ = 0,21 тыс. у.е.

Суточный экономический эффект $Э_{сут} = (2-С_п)*4-З_о$

где $С_п$ – условно-переменные затраты, $С_п$ = 0,1 тыс. у.е.

Полный экономический эффект $Э_т = Э_{сут} * 365 / (К_р + К_{эн})$.

ГЛАВА 7. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Для программирования сложных алгоритмических структур таких, например, как ветвления и циклы, необходимо иметь представление о структурных операторах. *Структурные операторы* представляют собой структуры, построенные из других операторов по строго определенным правилам. Все структурные операторы подразделяются на три группы: *составные, условные и повтора.*

7.1. Составные операторы

Составной оператор представляет собой группу из произвольного числа операторов, отделенных друг от друга точкой с запятой и ограниченную операторными скобками `Begin ... End` [3].

Формат

```
begin
     <оператор;>
     <оператор;>
     . . .
     <оператор>
end;
```

Пример

Записать, используя составной оператор, выражение $R := \sin(\pi \cdot x) + e^{2x}$.

```
Begin
  A:= Sin(Pi*x);
  B:= exp(2*x);
  Rez:= A + B;
end;
```

Составной оператор может находиться в любой части программы и воспринимается как единое целое. Обычно используется при организации разветвлений или циклов.

7.2. Условные операторы

Условные операторы используются для реализации разветвляющихся алгоритмов.

Разветвляющимся называется алгоритм, который содержит несколько ветвей, отличающихся друг от друга содержанием вычислений. Выход вычислительного процесса на ту или иную ветвь алгоритма определяется исходными данными задачи [9].

В разветвляющихся вычислительных процессах отдельные этапы вычислений (операторы) выполняются не всегда в одном и том же порядке. В зависимости от некоторых условий, проверяемых по ходу вычислений, выбираются для исполнения различные их последовательности. Поэтому в программе должно содержаться указание о том, в каком случае надо выбирать для исполнения тот или иной оператор. Это указание естественно формулировать в виде отношения, например, условия $x > y$. Если это условие справедливо (принимает значение TRUE), то выполняется соответствующий этому условию оператор, в противном случае, если условие принимает значение FALSE, выполняется другой оператор.

Для задания подобного рода разветвляющихся вычислительных процессов служат условные (выбирающие) операторы, которые относятся к числу производных операторов. Такой оператор обеспечивает выполнение или невыполнение некоторого оператора, группы операторов или блока в зависимости от заданных условий. Pascal допускает использование двух условных операторов `IF` и `CASE`.

Оператор условия `IF` является одним из самых популярных средств, изменяющих естественный порядок выполнения операторов программы. Он может принимать одну из следующих форм:

полная форма

```
If <условие> Then <оператор1>
   Else <оператор2>;
```

Если условие истинно, выполняется оператор1, в противном случае – оператор2;

неполная форма

```
If <условие> Then <оператор>;
```

Если условие истинно, выполняется оператор, стоящий за `Then`, в противном случае – выполняется оператор, стоящий сразу за `If`.

Условие может быть простым и сложным. В последнем случае используются логические операции AND, OR, NOT.

Блок-схемы полной и неполной условной конструкции подробно рассмотрены в главе 4 на рис.4.3.

Пример

```
If X>0 Then Writeln('X больше 0')
       Else Writeln ('X меньше или равно 0');
```

Перед `Else` точка с запятой не ставится. Если после `Then` или `Else` необходимо выполнить не один, а несколько операторов, они оформляются как составной оператор, т.е. должны быть взяты в операторные скобки `Begin ... End`.

Пример

```
If X<=A Then Begin
                Y:= A*X;
                Z:= Ln(X)+Y;
             End;
```

Один оператор IF может входить в состав другого оператора IF. Такие конструкции называют вложенными.

Рассмотрим использование операторов условия для программирования разветвляющихся вычислительных процессов.

Пример 7.1.

Составить блок-схему и программу на языке Паскаль для вычисления значения функции Y:

$$y = \begin{cases} \log_2 |x - 4{,}2|, & x < 1{,}5 \\ \sin(2x), & x > 1{,}5 \end{cases}$$

Здесь имеет место простое логическое условие, так как разветвляющийся вычислительный процесс имеет только две ветви.

Первую ветвь условно обозначим Y1, вторую – Y2.

Блок-схема вычисления значения функции Y представлена на рис. 7.1.

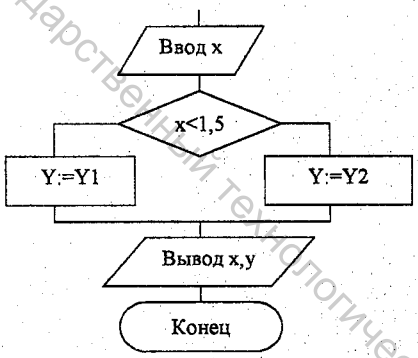


Рис.7.1. Блок-схема вычисления значения функции Y

Программа вычисления значения функции Y на языке Pascal:

```
Program razv1;
 uses crt;
 Var x,y:real;
 Begin
  Writeln('Введите x');
  Readln(x);
    If x < 1.5 Then Y:=Ln(Abs(x-4.2))/Ln(2)
               Else Y:=Sin(2*x);
  Writeln('x=',x:4:1, '  y=',y:5:2);
  Delay(5000);
 End.
```

Результат: x= 1.2 y= 1.58; x= 3.4 y= 0.49

Пример 7.2.

Составить блок-схему и программу на языке Паскаль для вычисления значения функции Y:

$$Y=\begin{cases}1, & x<0\\ 2+\sin(2x+1), & 0\le x<3\\ -1, & x\ge 3\end{cases}.$$

Здесь разветвляющийся вычислительный процесс имеет три ветви.

Блок-схема решения задачи представлена на рис. 7.2.

Программа вычисления значения функции Y представлена ниже.

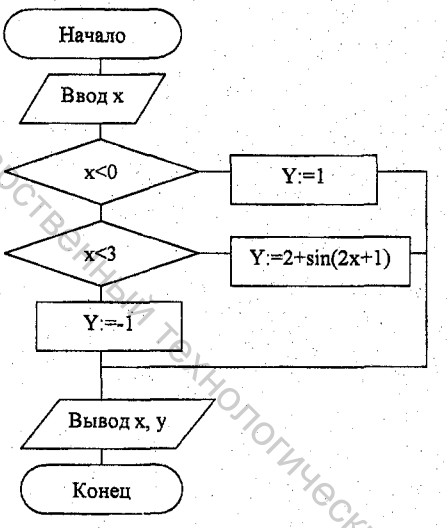


Рис.7.2. Блок-схема вычисления значения функции Y

```
Program razv2;
  uses crt;
 Var x,y:real;
 Begin
  Writeln('Введите x');
  Readln(x);
    If x < 0 Then Y:=1
                  Else If x < 3
                  Then Y:=2*Sin(2*x)
                  Else Y:=-1;
  Writeln('x=',x:4:1, '  y=',y:5:2);
  Delay(5000);
End.
```

Результат: x =-2.0 y = 1.00; x = 2.5 y =-1.92; x = 3.5 y= -1.00

Оператор выбора CASE является обобщением оператора IF и позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого *селектором*, и списка параметров, каждому из которых предшествует список констант выбора. Как и в операторе IF, здесь может присутствовать слово ELSE, имеющее тот же смысл. Блок-схема оператора Case представлена на рис. 7.3.

Формат

```
Case <выражение-селектор> of
     <список1>: <оператор1;>
     <список2>: <оператор2;>
     . . .
     <списокN>: <операторN>
     else <оператор>
end;
```

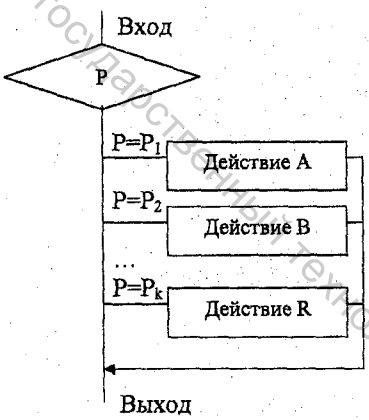


Рис. 7.3. Блок-схема оператора Case

Эта структура обеспечивает выбор одного из ряда возможных действий, в зависимости от значения, которое принимает выражение (условие) P. Количество значений, принимаемых P, должно быть конечным. Селектор P может быть любого скалярного типа, кроме вещественного.

Пример

Селектор целочисленного типа

```
Case I of
1:Z:=A+B;
2:Z:=A+10;
3:Z:=A+100
end;
```

Селектор перечисляемого типа

```
Case Season of
     Winter: Writeln('WINTER');
     Spring: Writeln('SPRING');
     Summer: Writeln('SUMMER');
     Autumn: Writeln('AUTUMN')
End;
```

Пример 7.3.

Составить программу, запрашивающую букву и выводящую название цветка, начинающееся на эту букву.

```
Program Flower;
Uses CRT;
 Var X:Char;
 Begin
  Writeln('Введите букву');
  readln(X);
  Case X of
  'A': Writeln('Астра');
  'B': Writeln('Бегония');
  'G': Writeln('Гвоздика');
  'R': Writeln('Роза');
  'V': Writeln('Василек');
  else Writeln('-----'); end;
  Delay(5000);
 End.
```

7.3. Задания для самостоятельной работы

Указания к решению. В заданиях вариантов 1 - 20 необходимо использовать оператор условия IF, в заданиях вариантов 21 - 30 – оператор выбора Case.

Вариант 1. Линейная плотность шерстяного волокна составляет 0,4 текс, а линейная плотность нитронового волокна равна 0,33 текс. Определить, возможно ли использование данных видов волокон в смеси.

При смешивании шерстяных волокон с химическими необходимо соблюдать условие $d_x \le d_ш$, где d_x и $d_ш$ – диаметры соответственно химического и шерстяного волокна. Диаметр волокна определяется по формуле

$$d = 35{,}7\sqrt{\frac{T}{\gamma}},$$

где T – линейная плотность волокна, текс;

γ - средняя плотность волокна, г/см^3.

Для шерстяного волокна γ=1,33 г/см^3, для нитронового волокна γ=1,17 г/см^3.

Вариант 2. Индекс ворсистости пряжи определяется по формуле

$$I = \frac{R_2(4R_4 + 3R_3 + 2R_2)}{R_1},$$

где R_1, R_2, R_3, R_4 – среднее число волокон длиной соответственно 1, 2, 3 и 4 мм (или немного более) в отрезке пряжи длиной 76 мм.

При I>10 пряжа считается ворсистой, в противном случае – неворсистой.

Определить качество пряжи, если а) R_1=2, R_2=1, R_3=4, R_4=3; б) R_1=3, R_2=2, R_3=2, R_4=1.

Вариант 3. Ранговый коэффициент корреляции (РКК) служит для предварительной оценки зависимости между двумя переменными величинами. Если $PKK \geq 0{,}5$, связь между переменными существенна и требуется более точное определение коэффициента корреляции.

$$PKK = 1 - \frac{6\sum \alpha^2}{n(n^2-1)},$$

где $\sum \alpha^2$ – квадрат суммарной разности значений рангов;

n – количество испытаний.

Определить, существенна ли связь между стойкостью немытой шерсти и волокон ровницы к истиранию и целесообразно ли вычисление коэффициента корреляции между ними, если n=11, $\sum \alpha^2 = 132$.

Вариант 4. Разработаны 2 варианта усовершенствования условий труда работников экспериментального цеха. Годовая экономия затрат по первому варианту $Э_1$ составит 5,5 млн. руб., по второму варианту $Э_2$=5,3 млн. руб. Стоимость приобретения средств механизации для первого варианта составляет $К_1$=24 млн. руб., для второго варианта – $К_2$=16 млн. руб. Определить, какой из двух вариантов механизации более эффективен.

Критерием эффективности механизации является коэффициент экономической эффективности, вычисляемый по формуле

$$E = \frac{Э}{K},$$

где Э – годовая экономия затрат, млн. руб.;

К – стоимость приобретения средств механизации, млн. руб.

Для решения задачи необходимо определить, для какого из вариантов усовершенствования коэффициент будет наибольшим.

Вариант 5. Выбрать наиболее эффективный вариант использования новой техники на предприятиях легкой промышленности и определить годовой экономический эффект его внедрения при одинаковом объеме и качестве продукции. Наиболее экономичным является тот вариант, где приведенные затраты наименьшие.

Годовой экономический эффект вычисляется по формуле

$$Э = (З_б - З_i)П,$$

где $З_б$ – приведенные затраты для базового варианта, тыс. руб.;

$З_i$ – наименьшие приведенные затраты по одному из вариантов, тыс. руб.;

П – объем производства, П=10000 единиц продукции в год.

Приведенные затраты для базового варианта $З_б$=1237,5 тыс. руб., для первого варианта $З_1$=1160 тыс. руб, для второго варианта $З_2$=1150 тыс. руб.

Вариант 6. Замена одного съемного барабана чесальной машины двумя дает гарантийное повышение производительности с сохранением качества продукта при выполнении двух условий:

$$Kc_2 > \frac{Kc}{2-Kc} \text{ и } Kc_1 \geq \frac{Kc_2}{1+Kc_2},$$

где Кс – коэффициент съема при наличии одного барабана;

$Кс_1$ и $Кс_2$ – коэффициент съема первого и второго барабанов соответственно.

Повышение производительности труда η в этом случае определяется по формуле

$$\eta = \frac{2Kc_2 - Kc \cdot Kc_2 - Kc}{2Kc_2 - Kc \cdot Kc_2 + Kc} \cdot 100.$$

Если условие не выполняется, η вычисляется по формуле

$$\eta = \frac{Kc_1 + Kc_2 - Kc_1 \cdot Kc_2 - Kc}{(1-Kc)(Kc_1 + Kc_2 - Kc_1 \cdot Kc_2) + Kc} \cdot 100$$

Определить, повысится ли производительность труда, если Кс=0,08, $Кс_1$=0,055, $Кс_2$=0,05.

Вариант 7. Определить качество прочеса химического волокна, если в 1 г продукта содержится 1 крупная мушка, 4 средних и 8 мелких мушек.

Мушка – закатанные в комочек волокна, не разделяющиеся без обрыва.

Для решения задачи перевести крупные и мелкие мушки в средние с учетом того, что мелкая мушка оценивается в 1/3 средней мушки, крупная — в 3 средние мушки. Допустимое число средних мушек в 1 г для чесальной ленты из химического волокна – 11, в противном случае прочес некачественный.

Вариант 8. Степень организации процесса получения чесальной ленты можно оценить по коэффициенту вариации чесальной ленты. Если коэффициент вариации ленты $CV \leq 4{,}8$, то процесс организован нормально, в противном случае в процесс получения чесальной ленты необходимо внести изменения.

Сделать вывод об организации процесса получения чесальной ленты при коэффициенте вариации, равном а) 2,2%; б) 4,38; в) 6%.

Вариант 9. Уменьшение численности работающих в результате усовершенствования технологического процесса составило $У_ч$=27 человек при численности работающих в базисном периоде $Ч_р$=300 человек и планируемом на год росте объема производства $Р_п$=8%. Определить, как изменится производительность труда и повлияет ли это на объем производства.

Изменение производительности труда

$$\Delta П_т = \frac{У_ч}{Ч_{пл} - У_ч} \cdot 100,$$

где $Ч_{пл}$ – условная численность в плановом периоде, $Ч_{пл} = Ч_р\left(1 + \frac{P_п}{100}\right)$;

Доля прироста объема производства за счет повышения производительности труда определяется по формуле

$$\Delta M = \frac{\Delta П_т}{P_п} \cdot 100 .$$

Если ΔМ>100%, то наблюдается прирост объема производства, в противном случае наблюдается спад объема производства.

Вариант 10. Неровнота, %, вносимая в пряжу самой прядильной машиной в ходе прядения, вычисляется по формуле

$$CV_м = \sqrt{CV_ф^2 - CV_{ид}^2} ,$$

где $CV_ф$ – фактическая неровнота пряжи по Устеру, %;
$CV_{ид}$ – идеальная неровнота с учетом условий прядения, %.

$$CV_{ид}^2 = CV_р^2 + 10{,}44\frac{(E-1)d^2}{T_р} ,$$

где $CV_р$ – неровнота ровницы по Устеру, %;
E – вытяжка в прядении;
T – линейная плотность ровницы, текс;
d – диаметр волокна, мкм.

При правильном ведении технологического процесса величина $CV_м$ не должна превышать 10%, в противном случае прядильная машина требует проверки и исправления.

Определить, правильно ли ведется технологический процесс, если $CV_ф$=19%, $CV_р$=5,5%, E=18, d=21 мкм, T=384 текс.

Вариант 11. Количество отходов в прядении (у) зависит от обрывности нити (х). Определить вид связи между этими переменными, если $P_{y/x}$=6,29, $P_{x/y}$=0,113.

Вид связи определяется по значению коэффициента корреляции r между переменными, который есть среднее геометрическое коэффициентов регрессии, имеющее их знак, т.е.

$$r = \pm\sqrt{P_{y/x} \cdot P_{x/y}} ,$$

где $P_{y/x}$ и $P_{x/y}$ – коэффициенты регрессии соответственно у относительно х и х относительно у.

Связь считается слабой, если $|r| < 0{,}3$, если $0{,}3 \le |r| < 0{,}7$ - связь средняя, при $0{,}7 \le |r| \le 1$ связь считается сильной.

Вариант 12. Соответствие образца шерсти норме определяется по коэффициенту относительной равномерности Q. При Q=1 образец шерсти соответствует норме, при Q<1 равномерность ниже нормы, т.е. имеется избыток коротких или длинных волокон, при Q>1 равномерность выше нормы, т.е. доля коротких или длинных волокон занижена и образец имеет повышенную равномерность.

Значение Q вычисляется по формуле

Для волокон со средней длиной до 75 мм включительно	Для волокон со средней длиной более 75 мм
$Q = \frac{B \cdot l}{2510 + 17l}$	$Q = \frac{Bl}{4800 - 11{,}5l}$

где B – наибольшая сумма плотностей пяти смежных классов волокон, %, а l – длина волокна, мм.

Определить, соответствует ли образец шерсти норме, если средняя длина волокна в образце равна 80 мм, а B=64%.

Вариант 13. Определить, будет ли закатываться шерсть в процессе промывки, если общий объем образца разрыхленной шерсти V=1000 см^3, средняя плотность волокна γ=1,33 г/см^3, масса сухой шерсти в общем объеме образца перед разрыхлением G=180 г.

Степень рыхления шерсти перед промывкой η, %, характеризуется уменьшением ее средней плотности

$$\eta = \left(1 - \frac{G}{V \cdot \gamma}\right) \cdot 100 .$$

Оптимальным считается $80 \le \eta \le 90\%$, если η>90%, то шерсть в процессе промывки будет закатываться, при η<80% промывка шерсти затрудняется.

Вариант 14. Определить состояние весового питателя, если суммарное поступление материала за первые 5 последовательных бросков S_1=1777 г, а суммарное поступление волокна за последующие 5 последовательных бросков S_2=1731 г.

При отвешивании порций волокнистого материала должна соблюдаться близость суммарных поступлений S материала за каждые 5 последовательных бросков. Для этого необходимо R по формуле

$$R = \frac{S_{max} - S_{min}}{S_{max}} \cdot 100 ,$$

где S_{min} и S_{max} – суммарное поступление волокнистого материала за первые и последующие 5 последовательных бросков (S_1 или S_2).

Если R<1%, весовой питатель находится в отличном состоянии, при $1\% \le R < 1{,}5\%$ весовой питатель находится в удовлетворительном состоянии, при R>1,5% состояние весового питателя неудовлетворительное и требуется его наладка.

Вариант 15. Определить степень загрузки швейного потока с последовательно-ассортиментным запуском, если суммарная удельная трудоемкость изготовления изделий γ и время выполнения сменного задания Vr равны а) γ=1, Vr=28800 с.; б) γ=1,01, Vr=29100 с.; в) γ=0,95, Vr=27400 с. Время смены R=28800 с.

Поток загружен нормально, если γ=1 и Vr=R, при γ>1 и Vr>R поток перегружен, т.е. рабочие работают с превышением допустимых норм, при γ<1 и Vr<R поток недогружен.

Вариант 16. Укорочение волокон в процессе чесания, %, определяется по формуле

$$\eta = \frac{l_1 - l_2}{l_1} \cdot 100,$$

где l_1 и l_2 – средняя длина волокна, мм, до и после чесания соответственно.

Критерии оценки степени укорочения η

η	Оценка
≤ 15%	Отлично
15 < η ≤ 18%	Хорошо

η	Оценка
18 < η ≤ 21%	Удовлетворительно
> 21%	Неудовлетворительно

Определить степень укорочения волокон в процессе чесания, если l_1=65 мм, l_2=57 мм.

Вариант 17. Определить сорт хлопкового волокна, если расчетное содержание пороков и примесей, %, равно а) 2,7; б) 1,6; в) 5; г) 6,7.

Сорт хлопкового волокна определяется в соответствии с таблицей

Сорт	Предельно допустимое содержание пороков
О	3
I	4

Сорт	Предельно допустимое содержание пороков
II	5,5
III	7

Вариант 18. Определить вид шерсти, используемой для смешивания с вискозным волокном, если линейная плотность вискозного волокна равна а) 0,34; б) 0,41; в) 0,55 текс.

Для определения вида шерсти воспользуйтесь таблицей

Вид шерсти	Линейная плотность вискозного волокна, текс
Тонкая	0,33-0,4
Полутонкая	0,4-0,45

Вид шерсти	Линейная плотность вискозного волокна, текс
Неоднородная	0,45-0,83

Вариант 19. Определить, как пойдет процесс прядения, если фактическая линейная плотность $T_ф$ равна 41 текс, а расчетная линейная плотность $T_р$ равна а) 40 текс; б) 41 текс; в) 48 текс.

Оценить процесс прядения можно по резерву (запасу) прядильной способности шерсти J, %

$$J = \left(1 - \frac{T_p}{T_ф}\right) \cdot 100.$$

Если J=0, процесс прядения пойдет плохо, при J<0 процесс пойдет очень плохо, при J>0 процесс пойдет хорошо.

Вариант 20. Известны $\bar{x}$ - среднее арифметическое значение объема производства трикотажных изделий, млн. шт., предприятиями Республики Беларусь, x' - среднее из тех значений выпуска продукции, которые меньше $\bar{x}$, x'' - среднее из тех значений выпуска продукции, которые больше $\bar{x}$. Определить эксцесс и сделать вывод о смещении кривой распределения значений выпуска трикотажной продукции, если $\bar{x}$=26, x'=20,42, x''=32,01, σ=2,21.

Эксцесс рассчитывается по формуле

$$\varepsilon = 1 - \frac{\sqrt{2\pi}(x'' - \bar{x}) \cdot (\bar{x} - x')}{\sigma(x'' - x')},$$

где σ - среднее квадратическое отклонение.

Если $\varepsilon > 0$, то кривая лежит выше кривой нормального распределения, если $\varepsilon < 0$ - кривая лежит ниже кривой нормального распределения, $\varepsilon = 0$ - совпадает с кривой нормального распределения.

Вариант 21. Составить программу для определения стоимости 1 секунды работы со 2 по 6 разряд, если стоимость 1 секунды работы 1 разряда равна 3 рубля. Тарифные коэффициенты для каждого разряда представлены в таблице.

Разряд	2	3	4	5	6
Тарифный коэффициент	1,56	2,03	2,16	2,27	2,5

Стоимость 1 секунды работы i-го разряда = Стоимость 1 с. 1 разряда * Тарифный коэффициент.

Вариант 22. На предприятии установлен повышающий коэффициент за стаж работы в отрасли в соответствии с таблицей. Составить программу для определения оклада сотрудника, отработавшего определенное количество лет.

Коэффициент	1,1	1,2	1,3	1,4	1,7	2
Стаж	0-3	4-6	7-10	11-15	16-20	>20

Оклад = Тарифная заработная плата * Повышающий коэффициент.
Значение тарифной заработной платы задать самостоятельно.

Вариант 23. В соответствии с коллективным договором между администрацией и профсоюзным комитетом на предприятии установлена премия работникам, имеющим детей. Размер премии устанавливается в соответствии с таблицей.

Количество детей	1	2	3	>3
Премия, %	10	20	30	50

Составить программу для вычисления размера премии сотрудников, имеющих детей. Значение оклада задать самостоятельно.

Премия = Оклад * % премии / 100.

Вариант 24. При расчете заработной платы рабочего бригады учитывается коэффициент трудового участия рабочего, который зависит от отработанного времени в соответствии с таблицей.

Отработанное время	100-149	150-199	200-249	250-299	300-349	350-400
КТУ	1,2	1,3	1,4	1,5	1,6	1,7

Составить программу для определения расчетной заработной платы рабочего бригады с учетом КТУ.

Расчетная заработная плата = Тарифная заработная плата * КТУ.

Значение тарифной заработной платы задать самостоятельно.

Вариант 25. Составить программу для определения розничной цены указанного вида товара, если магазином устанавливается торговая наценка в размере 25% от отпускной цены. Отпускные цены товаров представлены в таблице.

Шифр товара	1534	2246	1468	1132	1652
Отпускная цена	1520	11600	21300	850	22100

Розничная цена рассчитывается по формуле

Розничная цена = (100 + наценка) / 100 * отпускная цена

Вариант 26. Составить программу для определения размера начислений сотрудникам отдела продаж. Размер начислений зависит от количества продаж и вычисляется в соответствии с таблицей. Значение минимального размера начислений задать самостоятельно.

Продажи	От 15 до 25	От 26 до 30	От 31 до 50	От 51 до 75	От 76 до 100	>100
Коэффициент начислений	1,2	1,3	1,4	1,5	1,7	2

Начислено = Коэффициент начислений * минимальный размер начислений.

Вариант 27. Составить программу для определения тарифной заработной платы рабочих экспериментального цеха с учетом тарифного коэффициента.

Тарифная заработная плата = Минимальная базовая величина * Тарифный коэффициент.

Минимальная базовая величина = 62000 рублей.

Разряд	1	2	3	4	5	6
Тарифный коэффициент	1	1,15	1,54	1,85	2,31	3,08

Вариант 28. Составить программу определения расценки на выполнение каждой технологической операции в соответствии с разрядом работ. Стоимость

1 секунды выполнения операции по каждому разряду и затраты времени на выполнение операций представлена ниже.

Разряд	2	3	4
Стоимость 1 с., руб.	5,2	6	6,5

№ операции	1	2	3	4	5	6
Тарифный разряд	2	3	3	3	3	4
Затрата времени	24	117	50	38	80	92

Расценка на выполнение операции = Стоимость 1 секунды * Затрата времени

Вариант 29. Составить программу для определения оптовой стоимости партии товаров каждого вида на основании данных из таблиц 7.1 и 7.2.

Оптовая стоимость = оптовая цена * количество пар.

Таблица 7.1. Ведомость

Код товара	Месяц закупки	Количество пар
301	1	20
201	2	10
401	3	30
301	4	40
201	5	10
301	6	25
201	7	5
401	8	45

Таблица 7.2. Цены

Код товара	Оптовая цена 1 пары
301	26000
201	74000
401	45000

Вариант 30. Составить программу для определения экологического налога на выбросы загрязняющих веществ (на основании данных из таблиц 7.3 и 7.4).

Сумма налога = Фактически сожжено топлива * Ставка налога * Понижающий коэффициент.

Таблица 7.3 Данные для расчета экологического налога

Вид топлива	Фактически сожжено топлива за отчетный месяц, $м^3$	Ставка налога за выбросы, руб.	Сумма налога к уплате, руб.
1	242	1155	
2	319	1128	
3	756	735	

Таблица 7.4 Понижающий коэффициент

Сожжено топлива за отчетный месяц, $м^3$	Понижающий коэффициент за передвижные источники выбросов
До 200	0,9
От 201 до 500	0,85
Более 500	0,7

ГЛАВА 8. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Цикл – это многократное использование одного и того же вычисления с изменяющимися исходными параметрами.

Конкретные программные реализации циклов называются циклическими программами.

Различают два типа циклов:

- циклы со счетчиком, когда число повторений цикла известно заранее;
- итерационные циклы, когда число повторений цикла не известно заранее. В этом случае задается условие, при котором цикл должен закончиться или, наоборот, продолжаться.

Можно выделить *основные типы задач*, которые сводятся к программированию циклических процессов:

1. *Вычисление суммы, произведения элементов конечного ряда.* При этом суть циклического процесса заключается в накоплении суммы S или произведения P по зависимости

$$S := S + U,$$

где S, стоящее справа от «:=», есть сумма элементов конечного ряда,
U – значение текущего элемента конкретного ряда,
S, стоящее слева от «:=», - последующее (определяемое) значение суммы конечного ряда.

Или

$$P := P*U,$$

где P, стоящее справа от «:=», есть произведение элементов конечного ряда,
U – значение текущего элемента конкретного ряда,
P, стоящее слева от «:=», - последующее (определяемое) значение.

2. *Определение суммы (произведения) элементов бесконечного ряда.* Условием окончания цикла является дополнительное логическое условие $|S_n - S_{n-1}|<\varepsilon$, где ε – заданная точность вычислений. Сумма при этом вычисляется традиционно: S:=S+U.
3. *Табулирование функции* – получение значений функции Y=f(x) на интервале $[X_{min}; X_{max}]$ при дискретном изменении значения аргумента dx.

X	X_{min}	$X_{min+\Delta X}$	. . .	X_{max}
Y	Y_1	Y_2	. . .	$Y=f(X_{max})$

Для реализации циклов в Паскале используются три оператора повтора:

`For` – для реализации циклов, число повторений которых известно заранее (циклах со счетчиком);

`Repeat` и `While` – используются как в итерационных циклах, так и в циклах со счетчиком.

8.1. Оператор повтора For

Оператор повтора `For` состоит из заголовка и тела цикла. Он может быть представлен в двух форматах:

```
For <параметр цикла>:=N1 to N2 do <оператор>;
For <параметр цикла>:=N1 downto N2 do <оператор>;
```

здесь

`For ... do` – заголовок цикла;

`<оператор>` - тело цикла.

Оператор может быть простым или составным. В последнем случае используются операторные скобки.

Параметр цикла, его начальное и конечное значения должны принадлежать к одному и тому же типу данных. При этом допустим любой скалярный тип, кроме вещественного.

`N1` и `N2` – соответственно начальное и конечное значения параметра цикла.

Если используются типы `integer`, `byte` и интервальный, то параметр цикла в процессе выполнения циклической части программы принимает следующие значения:

в случае использования первого формата: N1, N1+1, N1+2,... N2;

в случае использования второго формата: N1, N1-1, N1-2,...N2.

Отсюда следует, что в первом случае (при `For ... to`) N1 должно быть меньше или равно N2, во втором (при `For ... downto`) N1 больше или равно N2. То есть шаг изменения параметра цикла в операторе `For` в любом случае равен единице. Однако это не является большим недостатком, так как любой шаг можно задать при использовании операторов `While` и `Repeat`.

Блок-схема оператора `For` представлена на рис. 8.1.

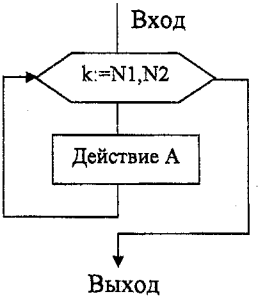


Рис.8.1. Блок-схема оператора For.

Эта конструкция обеспечивает повторение действия А заданное число раз. При каждом повторении цикла величина К, называемая параметром цикла, увеличивается (или уменьшается) на 1. Выход из цикла происходит тогда, когда величина К превзойдет значение N, которое определяет число повторений цикла.

После нормального завершения оператора For значение параметра цикла равно конечному значению. Если оператор For не выполнялся, значение параметра цикла не определено. В теле оператора For могут содержаться другие операторы For. Это позволяет строить циклы, содержащие внутренние циклы. Такие внутренние циклы называются *вложенными.*

Пример

```
For i:=4 to 7 do
    Write(i:2);
```

Результат: 4 5 6 7.

```
For i:=7 downto 4 do
Write(I:2);
```

Результат: 7 6 5 4.

Пример 8.1.

Вычислить сумму элементов конечного ряда:

$$S=\sum_{n=2}^{8}(-1)^{n+1}\cdot\frac{(x+1)^{2n}}{(n+1)!}\cdot\sin(n\cdot x)$$

Задачи такого типа сводятся к вычислению значения каждого элемента суммы при разных значениях параметра суммирования (в данном случае n). Каждое слагаемое суммы зависит от значения переменной X и параметра суммирования (параметра цикла) n, определяющего место этого слагаемого в сумме. Обычно формула общего члена суммы принадлежит к одному из следующих трех типов:

а) $\frac{x^n}{n!};\ \ (-1)^n\frac{x^{2n+1}}{(2n+1)!};\ \ \frac{x^{2n}}{(2n)!};$

б) $\frac{\cos nx}{n};\ \ \frac{\sin(2n-1)x}{2n-1};\ \ \frac{\cos 2nx}{4n^2-1};$

в) $\frac{x^{4n+1}}{4n+1};\ \ (-1)^n\frac{\cos nx}{n^2};\ \ \frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$

В случае а) для вычисления суммы целесообразно использовать рекуррентные соотношения, т.е. выражать последующий член суммы через предыдущий. Это позволяет существенно сократить объем вычислительной работы. Кроме этого, вычисления члена суммы по общей формуле в ряде случаев невозможно (например, из-за наличия n!).

В случае б) применение рекуррентных соотношений нецелесообразно. Вычисления будут наиболее эффективными, если каждый член суммы вычислять по общей формуле.

В случае в) член суммы целесообразно представить в виде двух сомножителей, один из которых вычисляется по рекуррентному соотношению, а другой непосредственно. Например, в нашем случае, если

$Un=(-1)^{n+1}\cdot\frac{(x+1)^{2n}}{(n+1)!}\cdot\sin(nx)$, то полагаем

$C_n=(-1)^{n+1}, P_n=(x+1)^{2n},\ \ F_n=(n+1)!$ и вычисляем рекуррентно:

$C_n=-C_{n-1};\ \ p_n=p_{n-1}\cdot(x+1)^2;\ \ F_n=F_{n-1}\cdot(n+1),$

а $\sin(n\cdot x)$ вычисляем непосредственно. Таким образом, для члена суммы имеем

$U_n=C_n\cdot P_n/F_n\cdot\sin(nx)$

Алгоритм решения задач суммирования при значениях параметра суммирования, изменяющегося в некотором диапазоне (n = 2,8) с заданным шагом (в нашем случае шаг равен 1), сводится к циклу, когда при каждом прохождении цикла номер члена суммы изменяется на 1, а сумма изменяется на ее n-ый член, т.е.

$$S_n = S_{n-1} + U_n \qquad (8.1)$$

где S_n и S_{n-1} – сумма n и n-1 членов.

Формула (8.1) применяется многократно при n = 2, 3, 4, …, 8. При этом S1 = 0 (не просуммировано ни одного элемента), S – искомое значение суммы 8 элементов.

При составлении блок-схемы и программы нет необходимости использовать переменные с индексами S_n, U_n и т.п., поскольку одновременно в вычислениях участвуют лишь 2 значения: S_n и S_{n-1}, C_n и C_{n-1}, P_n и P_{n-1}, F_n и F_{n-1}. Конечным результатом является сумма всех элементов, и значения суммы одного, двух, трех и т.д. элементов запоминать не требуется. В таком случае можно использовать простые переменные, значения которых будут изменяться каждый раз при вычислении очередного элемента суммы. При этом в формулах $c := -c$, $p := p \cdot (x+1)^2$, $f := f \cdot (n+1)$ (блок 4) и S:=S+U (блок 5) переменные C, P, F и S справа и слева имеют разные значения: справа предыдущие (C_{n-1}, P_{n-1}, F_{n-1}, S_{n-1}), слева текущие (C_n, P_n, F_n, S_n).

Блок-схема алгоритма решения задачи 2 представлена на рисунке 8.2.

Ниже приводится программа вычисления суммы элементов ряда.

```
Program Cikl1;
(*Вычисление суммы заданного числа элементов ряда*)
Uses Crt; (*указание библиотечных модулей*)
Var (*раздел описания переменных*)
    X,P,U,S:Real;
    C,F,N:Integer;
Begin (*начало раздела операторов*)
Clrscr;
Writeln('Введите x');
Readln(x);
C:=1;
P:=Sqr(X+1);
F:=2;
S:=0;
(*начало цикла по N*)
For N:=2 to 8 do
  Begin
C:=-C;
P:=P*Sqr(X+1);
F:=F*(N+1);
U:=C*P/F*Sin(N*X);
S:=S+U;
  End;
```

```
    (*вывод результата на дисплей*)
    Writeln('При  X=',X:4:2,'     сумма',  N:2,'  элементов=',
S:7:5);
    Readln;
    End. (*конец программы*)
```

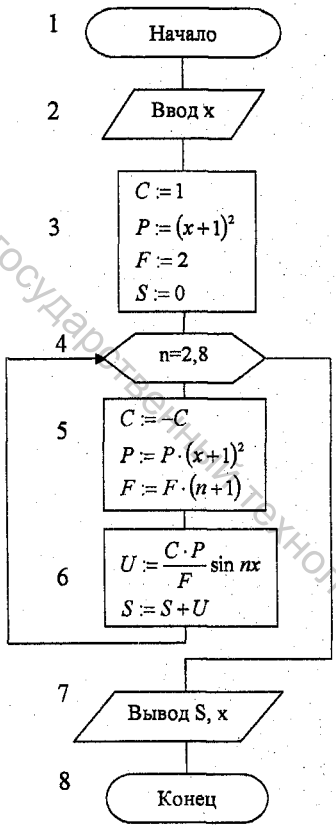


Рис. 8.2. Блок-схема вычисления суммы элементов ряда

8.2. Оператор повтора Repeat

Часто при программировании циклов заранее невозможно определить число необходимых итераций или приближений. Поэтому при решении таких задач применяется не цикл со счетчиком (или с параметром), а цикл с посту-

словием, называемый иначе итеративным, или *итерационным*. Первой особенностью такого цикла является то, что заранее неизвестно число его повторений.

Второй особенностью итерационного цикла является то, что тело цикла должно располагаться до проверки условия окончания (или повторения) цикла. В языке Паскаль для организации цикла с постусловием имеется специальный оператор `Repeat ... Until`, проверяющий условие окончания цикла.

Формат

```
Repeat
     <оператор;>
     <оператор;>
     . . .
     <оператор >
Until <условие>;
```

Условие – это выражение булевского типа, при котором цикл заканчивается. Операторы, заключенные между словами `Repeat` и `Until`, являются телом цикла. Вначале выполняется тело цикла, затем проверяется условие выхода из цикла. Если результат выражения `FALSE`, тело цикла выполняется еще раз, если результат `TRUE` – происходит выход из цикла.

Оператор `Repeat` имеет три характерные особенности:

- выполняется по крайней мере хотя бы один раз;
- тело цикла выполняется пока условие ложно;
- в теле может находиться произвольное число операторов без операторных скобок `begin ... end`.

По крайней мере, один из операторов тела цикла должен влиять на значение условия, иначе цикл будет выполняться бесконечно.

Блок-схема оператора повтора `Repeat` представлена на рис. 8.3.

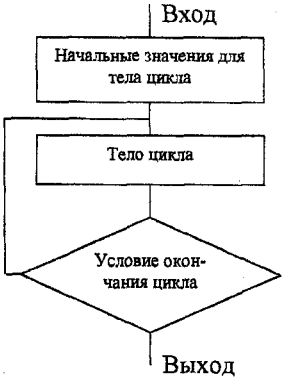


Рис. 8.3. Блок-схема оператора Repeat

Так как условие проверяется после выполнения тела цикла, оператор `Repeat` выполняется по крайней мере один раз. Поэтому важно четко проследить правильность вхождения в цикл и выхода из него. При неправильной организации выхода может также возникнуть ошибка переполнения, так как наращивание какой-либо величины в теле цикла может продолжаться до бесконечности.

Пример

```
i:=4;
Repeat
    Write(I:2);
        i:=i+1
Until I>7;
```

Результат: 5, 6, 7, 8

Пример 8.2.

Составить с объяснен $S = \sum_{n=1}^{\infty} \frac{\sin^n(x+1)}{n!} \cdot \left(\frac{x}{2}\right)^3$ у вычисления суммы элементов ряда с заданной т

$$S = \sum_{n=1}^{\infty} \frac{\sin^n(x+1)}{n!} \cdot \left(\frac{x}{2}\right)^3$$

Условием окончания цикла считать $|S_n - S_{n-1}| < \varepsilon$, где S_n – значение суммы элементов ряда, S_{n-1} – значение суммы n-1 элементов ряда, $\varepsilon = 10^5$ – заданная точность вычислений.

Очевидно, что формула в условии данной задачи относится к типу в).

Поэтому, полагая $P_n = P_{n-1} \cdot \sin(x+1)$, а $F_n = F_{n-1} \cdot n$, получим значение n-го элемента ряда $U_n = \frac{P_n}{F_n} \cdot \left(\frac{x}{2}\right)^3$ и суммы $S_n = S_{n-1} + U_n$. Блок–схема решения этой задачи представлена на рис. 8. 4. Дадим пояснения к блок-схеме.

Блок 1 – начало.

Блок 2 – ввод значений x, ε.

Блок 3 – задание начальных значений для организации цикла, (P:=1, F:=1) таких, что при N = 1, $P_n = \sin(x+1)$, а $F_n = 1!$. Очевидно, что нет необходимости повторять вычисление $(x/2)^3$ на каждом шаге цикла. Поэтому можно вычислить его заранее и поместить в переменную A. Так как вычисление суммы идет по стандартному алгоритму $S_n = S_{n-1} + U_n$, то начальное значение суммы обнуляется. Чтобы число просчитанных элементов ряда соответствовало значению N, N также принимается равным 0.

Блок 4 – образует тело цикла, здесь вычисляются значения P и F по рекуррентным соотношениям, приведенным выше, значения n-го элемента ряда U и суммы S. Так как U – есть не что иное, как $S_n - S_{n-1}$, нет необходимости для сохранения предыдущего (n-1) значения суммы вводить переменную S_1. Переменная DELTA сохраняет значение разности сумм на n-ом и (n-1) шаге как модуль U.

Блок 5 – проверка условия окончания цикла. Если оно истинно, т.е. $|U| < \varepsilon$, то осуществляется переход к блоку 5, если ложно – к блоку 4, и тело цикла повторяется.

Блок 6 – печать результатов – S, N, X.

Блок 7 – конец программы.

Программа вычисления суммы ряда с заданной точностью приведена ниже.

```
Program Cikl2; (*вычисление суммы элементов бесконечного ряда*)
Uses Crt; (*указание библиотечных модулей*)
Var (*раздел описания переменных*)
  X,EPS,P,A,U,S,S1,Delta:Real;
                    F,N:Integer;
Begin (*начало раздела операторов*)
Clrscr;
Writeln('Введите X');
```

```
    Readln(X);
    Writeln('Введите точность вычислений EPS');
    Readln(EPS);
    A:=exp(3*ln(x/2)); P:=1; F:=1; N:=0;S:=0;
    (*начало итерационного цикла*)
    Repeat
    N:=N+1; P:=P*Sin(X+1); F:=F*N;
    U:=P/F*A; S:=S+U;Delta:=Abs(U);
    (*проверка условия окончения цикла*)
    Until Delta<EPS;
    (*вывод результата на дисплей*)
    Writeln('При   x=',x:4:2,'      сумма',   n:2,'   элемен-
тов=',S:7:5);
    Readln;
    End.
    (*конец программы*)
```

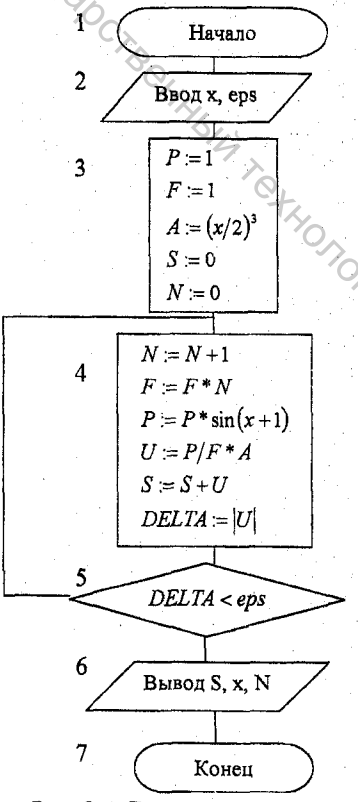


Рис. 8.4. Блок-схема вычисления суммы S

8.3. Оператор повтора While

Формат

```
While <выражение> Do <оператор>;
```

Этот оператор аналогичен оператору `Repeat`, но проверка условия выполнения тела цикла производится в самом начале оператора.

Блок-схема оператора `While` представлена на рис. 8.5.

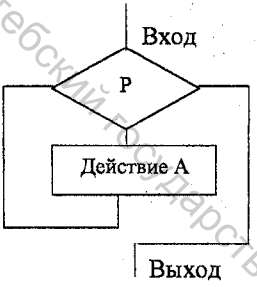


Рис.8.5. Блок-схема оператора While

Эта структура обеспечивает многократное повторение некоторого действия A (называемого телом цикла). Действие повторяется до тех пор, пока выполняется условие P (условие повторения цикла). Если после очередного повторения действия A условие P не выполняется, происходит выход из цикла.

Любой цикл, построенный с помощью `For` или `Repeat`, может быть построен и с помощью `While`. Здесь выражение представляет собой условие, при котором цикл повторяется. Перед каждым выполнением тела цикла вычисляется значение выражения условия. Если результат равен `True`, тело цикла выполняется и снова вычисляется выражение условия. Если результат равен `False`, то есть условие не выполняется, происходит выход из цикла и переход к первому после `While` оператору. Если перед первым выполнением цикла значение выражения было `False`, тело цикла вообще не выполняется и происходит переход к следующему оператору.

Цикл, который был описан для оператора REPEAT, в случае с WHILE будет иметь вид

```
i:=4;
While i<=7 do
    Begin
        Write(i:2);
    i:=i+1
End;
```

Как и в операторе `Repeat`, программист сам должен позаботиться об изменении переменных, определяющих условие выхода, иначе цикл получится

бесконечным. Выйти из цикла можно с помощью оператора `goto`, минуя вычисление выражения условия.

Для начинающих программистов использование оператора `While` вызывает гораздо меньшее число ошибок, чем применение оператора `Repeat`, поэтому при прочих равных условиях лучше пользоваться оператором `While`.

Таким образом, можно выделить следующие отличия `While` от `Repeat`:

1. Условие ставится в начале цикла.
2. Это условие, при котором цикл повторяется.
3. Необходимость использования операторных скобок `Begin... End`, если в цикле выполняется более одного оператора.
4. Выход из цикла будет произведен тогда, когда условие, стоящее после `While`, не будет выполняться.

Пример 8.3.

Разработать схему и программу на языке Паскаль задачи табулирования функции

$$Y = \begin{cases} \sqrt{t + \ln^2 |x|}, & x < 1 \\ t \cdot \sin^2(x+1), & 1 \le x \le 2{,}3 \\ \sqrt[3]{e^{2x+t}}, & x > 2{,}3 \end{cases}$$

заданной на отрезке [a, b], шаг приращения аргумента dx.

Для контрольного примера взять следующие значения исходных данных:

- границы интервала `a = -0,6; b = 3,4;`
- коэффициент `t = 1,2;`
- шаг изменения параметра цикла `dx = 0,25`.

Алгоритм решения этой задачи включает в себя следующие этапы:

- ввод исходных данных, в качестве которых выступают `a, b` – границы интервала табулирования функции, `dx` – приращение аргумента и коэффициент `t`;
- задание начального значения аргумента функции `x: = a`;
- организация цикла с помощью структуры повтор (цикл-пока), которая реализуется в программе оператором `While`.

Тело цикла состоит из следующих действий:

- вычисление значения функции, которая реализуется управляющей структурой развилка;
- печать значений аргумента и функции;
- приращение значения аргумента `x` на величину `dx`;
- после выполнения тела цикла управление передается на начало цикла, где проверяется условие повторения цикла. Если оно выпол-

няется, тело цикла повторяется, в противном случае происходит выход из цикла.

Блок-схема алгоритма решения задачи приведена на рис. 8.6.

В схеме необходимо обратить внимание на следующее:

- в блоке 4 организована проверка условия повторения цикла оператора `While`;
- в блоках 5 и 7 проверяют, в каком диапазоне значений находится величина аргумента `x` (для выбора соответствующей формулы для вычисления `y`) и реализуется в программе оператором `If`.

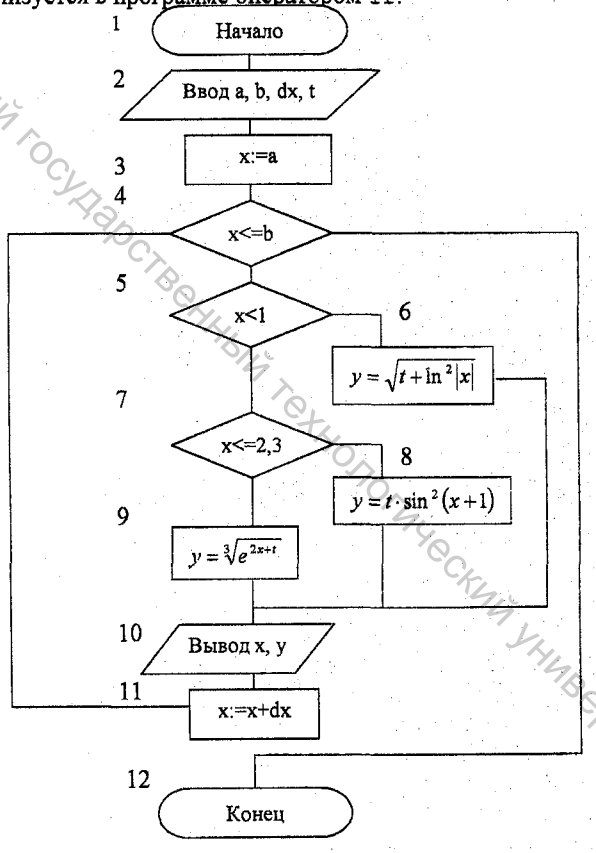


Рис.8.6. Блок-схема табулирования функции Y=f(x)

Текст программы решения задачи на языке Pascal приведен ниже.

```
Program Tab;
Uses crt;(*указание библиотечных модулей*)
 Var a,b,dx,t:real;
```

```
          x,y:real;
Begin (*начало раздела операторов*)
Clrscr;
Writeln('введите границы интервала [a,b],шаг dx-');
Readln(a,b,dx);
Writeln('Введите Значение t');
Readln(t);
(*вывод заголовка таблицы значений функции на дисплей*)
Write('Таблица значений функции');
Writeln;
x:=a; (*начальное значение a*)
While x<=b do
 Begin
(*вычисление и вывод y в зависимости от условия*)
 If x<1 Then y:=sqrt(t+sqr(ln(abs(x))))
        Else If x<=2.3 then y:=t*sqr(sin(x+1))
        Else y:= exp(1/3*(2*x+t));
(*печать таблицы значений функции на дисплей*)
 Writeln (X:5:2,' ':5,Y:5:2);
(*получение нового значения аргумента*)
 x:=x+dx;
 End; (*конец цикла*)
 Readln;
 End.
```

Рассмотренные операторы дают возможность описывать алгоритмы решения различных задач, которые представляют собой комбинацию линейных, разветвляющихся и циклических участков.

Линейные участки, которые не содержат разветвлений и циклов, реализуются обычно с помощью оператора присваивания, операторов ввода-вывода.

Разветвления в программе реализуются с помощью оператора `If`, а циклические участки – с помощью одного из операторов цикла – `For`, `While` и `Repeat`. Использование оператора `goto` для этих целей нежелательно, так как это нарушает логику программы/

Пример 8.4.

Для каждого из десяти задаваемых вещественных чисел вычислить значение Y по правилу

$$Y = \begin{cases} a\cos 2x, & x < 1 \\ a\sqrt{4 - \lg x}, & x \geq 1 \end{cases}$$

Программу можно представить в довольно наглядной и легко понимаемой форме:

```
Program Primer;
Uses crt;
Const n=10;
      a=2.5;
```

```
Var i: integer;
    x,y:real;
    Begin
    ClrScr;
      Writeln('Введите 10 чисел - значений x');
      For i:=1 to n do
      Begin
        read(x);
        if x<1 then y:=a*Cos(2*x)
                  else y:=a*sqrt(4-ln(x)/ln(10));
        Writeln('x=',x:4:1,'   y=',y:5:2);
        End;
       Delay(5000);
      End.
```

Результат:

Введите 10 чисел - значений x

2

x= 2.0 y= 4.81

5

x= 5.0 y= 4.54

8.4. Задания для самостоятельной работы

- **Вычислить значения функции на заданном интервале**[1]

Указание к решению. Программные реализации[2] заданий вариантов 1–10 требуют использования одного из операторов цикла: WHILE, REPEAT, FOR, вариантов 11 – 20 – использования оператора цикла (WHILE, REPEAT или FOR) и оператора условия IF.

Вариант 1. Изменение номинальной среднемесячной заработной платы описывается следующим полиномом

$y = 6{,}9536x^2 + 28{,}819x + 28{,}22$,

где x – период.

Найти значения среднемесячной заработной платы за период с 1995 по 2010 год.

[1] Функциональные зависимости y=f(x), предлагаемые для табулирования во всех вариантах рассчитаны на основании данных, приведенных в [9].

[2] При построении таблицы значений функции Y рекомендуется в качестве аргумента x использовать номер рассматриваемого периода. Например, если временной интервал - это годы 1999, 2000, ... , 2005, то 1999 году соответствует период 1, 2000 году – период 2, 2005 году – период 7 и т.д..

Вариант 2. Среднегодовую численность производственно-промышленного персонала по Республике Беларусь можно описать следующим полиномом четвертой степени

$$y = 0{,}3167x^4 - 7{,}4975x^3 + 57{,}498x^2 - 166{,}48x + 1285{,}5,$$

где x – период.

Найти среднегодовую численность производственно-промышленного персонала за период с 1995 по 2005 год.

Вариант 3. Производство химических волокон и нитей по Республике Беларусь в зависимости от года выпуска можно описать полиномом пятой степени

$$y = -0{,}0015x^5 + 0{,}1582x^4 - 3{,}3545x^3 + 26{,}393x^2 - 78{,}186x + 263{,}95,$$

где x – период.

Определить, сколько тонн химических волокон и нитей выпущено предприятиями Республики Беларусь за период с 1995 по 2005 год.

Вариант 4. Критическое число циклов движения волокон в чесальной машине вычисляется по формуле

$$z = \frac{\lg n \cdot m + \lg K_c}{-\lg(1 - K_c)},$$

где n – содержание волокон в 1 г, m – масса пучка, поступившего в чесальную машину, K_c – коэффициент съема.

Полное число циклов вычисляется по формуле

$$z_П = z + 1/K_c.$$

Определить критическое и полное число циклов движения волокон в чесальной машине при массе пучка от 0,04 до 1,4 г, если в 1 г продукта содержится 20000 волокон, а коэффициент съема равен 0,03.

Вариант 5. Интегральный уровень образования описывается следующей зависимостью

$$y = 0{,}0037\ln(x) + 12{,}567,$$

где x – период.

Определить интегральный уровень образования в Республике Беларусь в 2000-2015 г.г.

Вариант 6. Численность студентов на 10 тыс. человек населения описывается следующей зависимостью

$$y = 97{,}857x^{0{,}2098},$$

где x – период.

Определить численность студентов в 2000 – 20015 г.г.

Вариант 7. Объем производства чулочно-носочных изделий, млн. пар, предприятиями Республики Беларусь в зависимости от года выпуска можно описать полиномом пятой степени

$$y = -0{,}0287x^5 + 0{,}9751x^4 - 12{,}007x^3 + 63{,}377x^2 - 126{,}48x + 130,$$

где x – период.

Определить, сколько млн. пар чулочно-носочных изделий произведено предприятиями Республики Беларусь за период с 1995 по 2005 год.

Вариант 8. Индекс производства непродовольственных товаров в зависимости от года выпуска можно описать полиномом пятой степени

$$y = 0{,}0082x^5 - 0{,}3758x^4 + 6{,}2266x^3 - 46{,}069x^2 + 146{,}86x - 36{,}836,$$

где x – период.

Определить индексы производства непродовольственных товаров за период с 1995 по 2005 год.

Вариант 9. Доля гребенного очеса Q, %, в зависимости от протяженности зоны сортировки L, мм, может быть вычислена по формуле

$$Q = 0{,}03L^2 - 1{,}04L + 17{,}6.$$

Определить долю гребенного очеса при протяженности зоны сортировки от 20 до 30 мм. Шаг изменения протяженности зоны сортировки dL=1 мм.

Вариант 10. Коэффициент крутки $\alpha_т$ аппаратной пряжи из полугрубой и грубой шерсти рассчитывается по формуле

$$\alpha_m = (300 + 2T)/\sqrt{T},$$

где T – линейная плотность пряжи, текс.

Определить коэффициент крутки для аппаратной пряжи линейной плотности от 50 до 350 текс. Шаг изменения линейной плотности dT=50 текс.

Вариант 11. Индекс производства непродовольственных товаров в зависимости от года выпуска можно описать следующей зависимостью:

$$y = \begin{cases} 2{,}1963x^3 - 29{,}447x^2 + 119{,}27x - 22{,}167, & x \le 2000 \\ -3{,}3x + 111{,}3, & 2000 < x \le 2002, \\ -5{,}2x^2 + 20{,}3x + 94{,}1, & x > 2002 \end{cases}$$

где x – период.

Определить индексы производства непродовольственных товаров за период с 1995 по 2005 год.

Вариант 12. Индекс общего объема продукции легкой промышленности в зависимости от года выпуска можно описать следующей зависимостью:

$$y = \begin{cases} -12{,}575x^2 + 81{,}465x - 2{,}725, & x \le 1998 \\ -9{,}1212\ln x + 111, & 1998 < x \le 2002, \\ -5{,}65x^2 + 22{,}95x + 87{,}7, & x > 2002 \end{cases}$$

где x – период.

Определить индексы общего объема продукции легкой промышленности за период с 1995 по 2005 год.

Вариант 13. Индекс общего объема продукции машиностроительной отрасли в зависимости от года выпуска можно описать следующей зависимостью:

$$y=\begin{cases}23x+56{,}333, & x\le 1997\\ -1{,}1643x^2+4{,}4957x+112{,}1, & 1997<x\le 2002\\ -10{,}8x^2+46{,}1x+73{,}6, & x>2002\end{cases},$$

где x – период.

Определить индексы общего объема продукции машиностроительной отрасли за период с 1995 по 2005 год.

Вариант 14. Зависимость рентабельности реализованной промышленной продукции от года реализации продукции можно описать следующей зависимостью:

$$y=\begin{cases}8{,}5288e^{0{,}1366x}, & x\le 1999\\ 2{,}25x^2-11{,}65x+25{,}2, & 1999<x\le 2002\\ -1{,}6x^2+8{,}1x+5{,}5, & x>2002\end{cases},$$

где x – период.

Найти рентабельность реализации промышленной продукции за период с 1995 по 2005 год.

Вариант 15. Среднегодовую численность производственно-промышленного персонала по Республике Беларусь можно описать следующей зависимостью:

$$y=\begin{cases}1537x^{-0{,}3862}, & x\le 1995\\ -1{,}8333x^3+12x^2-7{,}1667x+1110, & 1995<x\le 1999\\ -1{,}2708x^4+18{,}838x^3-89{,}715x^2+129{,}46x+1092{,}8, & x>1999\end{cases},$$

где x – период.

Найти среднегодовую численность производственно-промышленного персонала за период с 1995 по 2005 год.

Вариант 16. Объем выпуска продукции в сопоставимых ценах в процентах к предыдущему году можно описать следующей зависимостью:

$$y=\begin{cases}76{,}392e^{0{,}1484x}, & x\le 1997\\ 0{,}1429x^2-2{,}8771x+115{,}24, & 1997<x\le 2002\\ -7{,}1x^2+30{,}1x+84{,}1, & x>2002\end{cases},$$

где x – период.

Определить объемы выпуска продукции в сопоставимых ценах за период с 1995 по 2005 год.

Вариант 17. Объемы производства химических волокон и нитей по Республике Беларусь в зависимости от года выпуска можно описать следующей зависимостью:

$$y=\begin{cases}23{,}9x^2-101x+287{,}7, & x\le 1997\\ -6{,}5667x^3+49{,}65x^2-102{,}18x+255{,}7, & 1997<x\le 2001,\\ 0{,}6833x^3-2{,}85x^2+2{,}0667x+204{,}4, & x>2001\end{cases}$$

где x – период.

Определить, сколько тонн химических волокон и нитей выпущено предприятиями Республики Беларусь за период с 1995 по 2005 год.

Вариант 18. Объем производства тканей, м^2, предприятиями Республики Беларусь в зависимости от года выпуска можно описать следующей зависимостью:

$$y=\begin{cases}-1120e^{-0{,}7853x}, & x\le 1995\\ -4{,}1667x^4+42{,}667x^3-144{,}83x^2+204{,}33x+135, & 1995<x\le 1999,\\ -1{,}6458x^4+20{,}051x^3-73{,}951x^2+83{,}066x+259{,}17, & x>1999\end{cases}$$

где x – период.

Определить объем производства тканей предприятиями Республики Беларусь за период с 1995 по 2005 год.

Вариант 19. Объем производства трикотажных изделий, млн. шт., предприятиями Республики Беларусь в зависимости от года выпуска можно описать следующей зависимостью:

$$y=\begin{cases}-1{,}6667x^3+16{,}286x^2-40{,}048x+64{,}6, & x\le 1999\\ -10{,}5x+69{,}667, & 1999<x\le 2002,\\ 2x^2-8x+47, & x>2002\end{cases}$$

где x – период.

Определить объем производства трикотажных изделий предприятиями Республики Беларусь за период с 1995 по 2005 год.

Вариант 20. Объем выпуска ковровых изделий, млн. м^2, предприятиями Республики Беларусь в зависимости от года выпуска можно описать следующей зависимостью:

$$y=\begin{cases}1{,}25x+3{,}05, & x\le 1999\\ 0{,}3667x^3-2{,}65x^2+4{,}6833x+6{,}3, & 1999<x\le 2003,\\ -1{,}15x^2+5{,}25x+2, & x>2003\end{cases}$$

где x – период.

Определить объем производства ковровых изделий предприятиями Республики Беларусь за период с 1995 по 2005 год.

- **Вычислить сумму элементов конечного ряда** (использовать оператор For).

Вариант	Задание
1.	$S=\sum_{n=1}^{8}(-1)^n\frac{x^{n+2}}{(n+1)n!}$
2.	$S=\sum_{n=1}^{9}(-1)^{n+1}\frac{x^{2n+1}\cdot n!}{2n+3}$
3.	$S=\sum_{n=1}^{8}(-1)^n\frac{x^{2n-1}\cdot\sin nx}{(2n+1)!}$
4.	$S=\sum_{n=2}^{6}\frac{x^{2n}\cdot\cos nx}{(n+2)!}$
5.	$S=\sum_{n=1}^{10}(-1)^{n+1}\frac{x^{n+2}}{(n+3)!}$

Вариант	Задание
6.	$S=\sum_{n=2}^{6}\frac{(x+0.5)^{2n}}{(2n-1)!}\cos nx$
7.	$S=\sum_{n=1}^{8}\frac{x^{2n+1}}{(2n+1)!}$
8.	$S=\sum_{n=1}^{8}(-1)^n\frac{x^{2n-1}}{(n+1)!}$
9.	$S=\sum_{n=1}^{10}\frac{(x+2)^{n+1}}{(n+1)!}x^3$
10.	$S=\sum_{n=1}^{8}(-1)^{n+1}\frac{x^{2n+2}}{(2n+2)!}$

- **Вычислить сумму элементов бесконечного ряда**

Вычислить сумму элементов бесконечного ряда с заданной точностью ε. Условием окончания вычислений считать $|S_n-S_{n-1}|<\varepsilon$. ε=0,00001.Значение x задать самостоятельно. Использовать оператор REPEAT.

Вариант	Задание
1.	$S=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{(2n+1)!}$
2.	$S=\sum_{n=1}^{\infty}\frac{x^{n+1}}{(n+1)!}$
3.	$S=\sum_{n=1}^{\infty}\frac{x^{2n+1}\cdot\sin nx}{2n!}$
4.	$S=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^{2n+1}}{n!}$
5.	$S=\sum_{n=2}^{\infty}(-1)^n\frac{x^n\cdot\sin^{n+1}(x+2)}{(2n+2)!}$

Вариант	Задание
6.	$S=\sum_{n=2}^{\infty}(-1)^n\frac{x^{n+1}}{(2n-1)!}$
7.	$S=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{(x+3)^{n+1}\cdot\sin^n x}{2n!}$
8.	$S=\sum_{n=1}^{\infty}(-1)^n\frac{x^n}{(n+1)!}$
9.	$S=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{\sin^{n+1}(x+2)}{(n+2)!}$
10.	$S=\sum_{n=1}^{\infty}(-1)^n\frac{x^{2n}}{(2n+1)!}$

ГЛАВА 9. МОДУЛЬНЫЕ ПРОГРАММЫ. ПОДПРОГРАММЫ ПОЛЬЗОВАТЕЛЯ

9.1. Понятие модульности в программировании

В практике программирования довольно часто встречаются ситуации, когда одну и ту же группу операторов, реализующих определенную цель, требуется повторить без изменений в нескольких других местах программы. Чтобы избавить программиста от столь нерационального занятия, была предложена концепция подпрограмм, впервые описанная М.Уилксом в 1957 году. Она получила широкое распространение во всех языках программирования. В соответствии с этой концепцией программа представляется в виде одного *глобального* модуля – *основной программы* и вложенных в нее модулей – *подпрограмм*.

Подпрограммой называется именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения по имени любое количество раз из различных мест программы.

В языке Pascal для организации подпрограмм используются *подпрограммы-процедуры* и *подпрограммы-функции*, которые в зависимости от целей программирования могут размещаться следующим образом:

- головная программа и подпрограммы распложены в одном программном модуле;
- подпрограммы расположены в отдельных файлах и включаются в основной файл посредством директивы компилятора;
- подпрограммы организуются как оверлейные структуры и поочередно загружаются в одно и тоже место памяти;
- подпрограммы пишутся на машинном коде и затем включаются в программу;
- подпрограммы располагаются во внешней библиотеке и вызываются из основной программы.

Все процедуры и функции языка Pascal разделяются на две группы: встроенные и описанные пользователем.

Встроенные (стандартные) процедуры и функции являются частью языка и могут вызываться по имени без предварительного определения в разделе описаний.

Паскаль представляет программисту достаточно широкий набор встроенных процедур и функций, которые реализуют наиболее часто встречающиеся при написании программ действия. Различают *9 основных групп* встроенных процедур и функций. Для обозначения типов данных аргументов используем сокращения: IBR – целочисленный, байтовый, вещественный тип; S – любой из скалярных типов, кроме вещественного.

1. *Арифметические* (Abs(IBR), Arctan(IBR), Cos(IBR), Sin(IBR), Exp(IBR)…).

2. *Скалярные* (Odd(I), Pred(S), Succ(S)).
3. *Преобразования типов* (Round(R) Trunc(R). Результат имеет целочисленный тип.
4. *Управление строками на экране* (ClrScr, DelLine, InsLine).
5. *Специальные функции* (Delay(I), Exit, Halt).
6. *Функции обработки строк.*
7. *Функции и процедуры обработки файлов.*
8. *Функции управления памятью.*
9. *Графические процедуры.*

Процедуры и функции пользователя организуются самим программистом в соответствии с синтаксисом языка и представляют собой локальный блок. Предварительное описание процедур и функций пользователя обязательно.

9.2. Процедуры пользователя

Использование подпрограмм, организованных пользователем, обеспечивает возможность:

- организовать работу нескольких программистов над одной программой;
- проводить отладку отдельных блоков и только потом всей программы в целом;
- значительно сэкономить память, т.к. многократно используемый участок заносится в память только один раз;
- упростить внесение изменений в программу.

Процедура пользователя – это именованная группа операторов, реализующая определенную часть общей задачи. Процедура вызывается по имени из любой части головной программы.

Описание процедуры включает заголовок и тело процедуры. Заголовок состоит из зарезервированного слова `Procedure`, идентификатора (имени) процедуры и необязательного списка формальных параметров, заключенного в круглые скобки. Для каждого параметра из списка указывается его тип.

Формат

```
Procedure <Имя процедуры>((формальные параметры);
{заголовок}
    <разделы описаний>
    Begin
      <раздел операторов>      тело процедуры
    End;
```

Пример заголовка

```
Procedure Sort(A:integer; B:real);
Procedure Sum;
```

Имя процедуры – это идентификатор, записанный по правилам записи идентификаторов и уникальный в пределах программы. Тело процедуры – это локальный блок, по структуре аналогичный программе.

Синтаксическая диаграмма процедуры представлена на рис. 9.1.

Рис.9.1. Синтаксическая диаграмма подпрограммы-процедуры

Для обращения к процедуре используется *оператор вызова процедуры*. Он состоит из идентификатора (имени) процедуры и списка фактических параметров, отделенных друг от друга запятыми и заключенных в круглые скобки.

Например, `SORT(A1,B1); SORT(14,25); SUM;` .

9.3. Параметры и переменные в модульных программах

Параметры подпрограмм. Параметры, описанные в подпрограммах, могут иметь любой тип. Следует различать *формальные* и *фактические* параметры.

Параметры, описанные в заголовке подпрограммы при ее определении, называются *формальными*. Например,

`Procedure Sort(A:integer; B:real)` – A и B – *формальные* параметры;

Параметры, которые указываются в заголовке подпрограммы при обращении к ней, называются *фактическими*. Например,

`SORT(A1,B1);` - фактические параметры – значения переменных;

`SORT(14,25);` - фактические параметры – непосредственные значения;

`SUM;` - фактические параметры не указаны, т.к. в вызываемой процедуре нет формальных параметров.

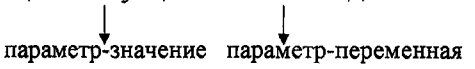
Когда параметры передаются по значениям, формальный параметр является переменной, локальной в блоке. Фактический параметр может быть любым выражением, переменной или идентификатором того же типа, что и соответствующий ему формальный параметр. Такие параметры называются *параметрами-значениями*. Их главная отличительная черта: изменение формального параметра не влечет за собой изменения фактического. Если же любое изменение формального параметра влечет за собой изменение фактического, то такие параметры называются *параметрами-переменными*. Для их описания служит зарезервированное слово `Var`.

Между фактическими параметрами в операторе вызова процедуры и формальными параметрами в заголовке описания процедуры устанавливается

взаимнооднозначное соответствие в результате их перебора слева направо. Количество и тип формальных параметров равны количеству и типу фактических параметров.

Если процедура возвращает в программу какие-то значения, соответствующие переменные должны быть описаны как параметры-переменные с использованием слова `Var`.

Например, `Procedure One(X:integer;  var Y:real);`



Таким образом, формальные параметры делятся на два типа: параметры-значения, которые передаются в подпрограмму, и параметры-переменные, которые передаются из подпрограммы в вызывающую часть.

Например,
```
Procedure One(X:integer; var Y:real);
          Begin
             Y:=X*Sin(X);
          End;
```
Вызов: `One(2.5, Z);`

Если в программе используется несколько процедур и одна из них вызывает другую, то вызываемая процедура должна быть описана обязательно до вызывающей.

Различные особенности описания процедур и способы их использования рассмотрим на примере следующей задачи [3].

Пример 9.1. По задаваемым вещественным значениям x и y вычислить `U = max(x+y, x*y), V = max(0.5, U).`

Алгоритм решения этой задачи сводится к поиску максимального значения из двух. Составим программу, реализующую решение без использования подпрограммы-процедуры.

```
Program Max_A;
    Var x,y,u,v:real;
  Begin
   Writeln('Введите x,y');
   Readln(x,y);
      If x+y>x*y  Then U:=x+y Else U:=x*y;
      If 0.5>U Then V:=0.5 Else V:=U;
      Writeln('U=',U:3:1,'  V=',V:3:1);
  End.
```

В программе фигурируют два условных оператора `If`, выполняющих одну и ту же частичную задачу: поиск максимального из двух значений. Поэтому алгоритм решения этой задачи целесообразно объявить процедурой.

Локальные и глобальные переменные. Программа на языке Pascal имеет модульную структуру и может состоять из ряда вложенных друг в друга блоков. Основная программа – это самый крупный блок, который не входит ни в какой другой. Те идентификаторы, которые описаны в головной программе, яв-

ляются *глобальными* и могут использоваться во всех вложенных блоках (во всех процедурах и функциях). Идентификаторы, описанные в определенных модулях, называются *локальными* и могут быть использованы только в той части программы, где они описаны. Все параметры, используемые в подпрограммах пользователя, подчиняются определенным правилам.

1. Каждый идентификатор должен быть описан перед тем, как он будет использован.
2. Областью действия идентификатора является блок, в котором он описан.
3. Идентификаторы в блоке должны быть уникальными, т.е. не должны повторяться.
4. Один и тот же идентификатор может быть по разному определен в каждом отдельном блоке.
5. Если процедура или функция пользователя совпадает по имени со стандартной процедурой или функцией, то будет выполняться подпрограмма пользователя, а стандартная игнорироваться (`Sin, Cos`).

Процедура без параметров. Обратимся к условию примера 9.1 и, чтобы подчеркнуть сходство двух условных операторов, запишем программу иначе:

```
... ... ...
    a:=x+y; b:=x*y;
    If a>b Then S:=a Else S:=b;
    U:=S;
    a:=0.5; b:=U;
    If a>b Then S:=a Else S:=b;
    V:=S;
... ... ...
```

Таким образом, условные операторы

```
If a>b Then S:=a Else S:=b;
```

полностью совпадают друг с другом, то есть один и тот же условный оператор присутствует в программе дважды. Объявим его процедурой.

```
Procedure Max_1A;
  Begin
     If a>b Then S:=a Else S:=b;
  End;
```

Для вызова этой процедуры в нужном месте программы достаточно записать оператор процедуры `Max_1A`.

Программа с этой процедурой выглядит следующим образом:

```
 Program Max_B;
  Var x,y,u,v,a,b,s:real;
(*******************************)
  Procedure Max_1A;
   Begin
    If a>b Then S:=a Else S:= b;
   End;
(*******************************)
```

```
Begin
   Writeln('Введите x,y');
   Readln(x,y);
   a:=x+y; b:=x*y;
   Max_1A; U:=S;
   a:=0.5; b:=u;
   Max_1A; V:=S;
   Writeln('U=',u:3:1,'  V=',v:3:1);
  End.
```

В этой программе переменные a, b, x, y, U, V, S описаны как глобальные и используются и в основной программе, и в подпрограмме-процедуре.

Процедура с параметрами. Предыдущий вариант не очень удобен, так как назначение процедуры зафиксировано слишком жестко. Следовательно, исходными данными для нее могут быть только значения переменных a и b. Поэтому при каждом обращении к процедуре этим переменным нужно присваивать те значения, из которых должно быть выбрано наибольшее. Этого можно избежать, оформив процедуру с параметром, и передавать значения a и b в качестве параметров-значений, а S – в качестве параметра-переменной.

```
Program Max_C;
  Var x,y,u,v:real;
(*************************************)
  Procedure Max_2A(a,b:real;var s:real);
   Begin
    If a>b Then S:=a Else S:= b;
   End;
(*************************************)
Begin
   Writeln(' Введите x,y');
   Readln(x,y);
   Max_2A(x+y,x*y,U);
   Max_2A(0.5,U,V);
   Writeln('U=',u:3:1,'  V=',v:3:1);
  End.
```

Здесь переменные x, y, U, V описаны как глобальные и, являясь фактическими параметрами при обращении к процедуре Max_2A, используются только в основной программе. Параметры a, b, S – формальные и используются при описании процедуры, причем, параметры a и b – параметры-значения, S – параметр-переменная.

9.4. Функции пользователя в языке Pascal

Хотя набор встроенных функций языка Pascal достаточно широк, он вряд ли может удовлетворить требованиям каждого программиста. Поэтому пользо-

вателю предоставлена возможность самому реализовывать нужные ему алгоритмы в виде функций и обращаться к ним из программ по мере необходимости. Функция, определенная пользователем, состоит из заголовка и тела функции. Заголовок состоит из ключевого слова Function, имени функции, необязательного списка формальных параметров, заключенного в круглые скобки с указанием типа каждого параметра, и типа возвращаемого результата.

Формат

```
Function <имя>[(формальные параметры)]:тип; - заголовок
    <разделы описаний>
    Begin
      <раздел операторов>        тело функции
    End;
```

Например, `Function One(X:integer):real;`
`Function Sum:real;`

В разделе операторов должен находиться по крайней мере один оператор, присваивающий идентификатору функции значение. Если таких операторов несколько, то результатом работы функции будет значение последнего оператора присваивания. Обращение к функции производится по имени, после которого в круглых скобках следует необязательный список фактических параметров. Синтаксическая диаграмма функции представлена на рис. 9.2.

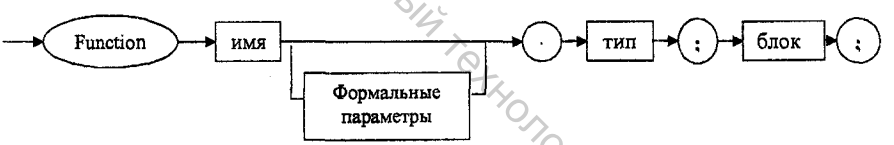


Рис.9.2. Синтаксическая диаграмма подпрограммы-функции

Имя функции – уникальный в пределах блока идентификатор. Раздел, относящийся к формальным параметрам для функции, аналогичен разделу описания формальных параметров для процедур. Возвращаемый результат может иметь любой скалярный тип.

Тело функции представляет собой локальный блок, по структуре аналогичный программе.

Особенности, присущие функциям:

1. Обязательно задается тип возвращаемого результата.
2. В результате работы функции получается только один результат.
3. В разделе операторов блока должен присутствовать хотя бы один оператор, влияющий на значение функции.
4. Обращение к функции должно происходить только из какого-либо выражения, *например*, `Y:=One(X);` а не `One(X)`.

Составим программу, реализующую решение рассмотренной выше задачи с использованием подпрограммы-функции.

Функция без параметров. Обратимся к условию примера 9.1, в котором необходимо вычислить наибольшее из двух значений U и V, где `U = max(x+y), V = max(0.5, U)`.

Основываясь на вышеприведенных рассуждениях, составим программу с использованием подпрограммы-функции без параметров.

```
Program Max_D;
  Var x,y,u,v,a,b:real;
(*************************************)
  Function Max:real;
   Var S:real;
   Begin
    If a>b Then S:=a Else S:= b;
    Max:=S;
   End;
(*************************************)
Begin
   Writeln('Введите x,y');
   Readln(x,y);
   a:=x+y; b:=x*y;
   U:=Max;
   a:=0.5; b:=u;
   V:=Max;
   Writeln('U=',u:3:1,'  V=',v:3:1);
  End.
```

В этой программе переменные `a, b, x, y, U, V` описаны как глобальные и используются и в основной программе, и в подпрограмме-функции `Max`, переменная `S` – локальная, так как используется только в подпрограмме-функции `Max`.

Функция с параметрами. Обратимся опять к условию примера 9.1, в котором необходимо вычислить наибольшее из двух значений U и V, где `U = max(x+y), V = max(0.5, U)`.

Основываясь на вышеприведенных рассуждениях, составим программу с использованием подпрограммы-функции с параметрами.

```
Program Max_E;
     Var x,y,u,v:real;
(*************************************)
     Function Max(a,b:real):real;
      Var S:real;
      Begin
       If a>b Then S:=a Else S:= b;
       Max:=S;
      End;
(*************************************)
     Begin
     Writeln(' Введите x,y');
```

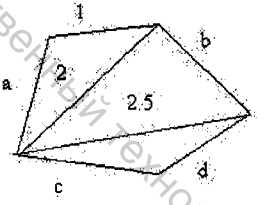
```
Readln(x,y);
U:=Max(x+y,x*y);
V:=Max(0.5,U);
Writeln('U=',u:3:1,'  V=',v:3:1);
End.
```

Здесь x, y, U, V – глобальные и, являясь фактическими параметрами подпрограммы-функции Max, используются только в основной программе. Параметры a, b – формальные параметры, используемые при описании подпрограммы-функции Max, переменная S – локальная, так как используется только в подпрограмме-функции Max.

9.5. Примеры описания подпрограмм при решении задач

Пример 9.2.

Вычислить площадь пятиугольника, изображенного на рисунке, где a,b,c,d – длины стороны (действительные числа).



Решение

Для вычисления площади пятиугольника достаточно вычислить площади треугольников со сторонами (a, 1, 2), (2, b, 2.5), (2.5, d, c), а затем их сложить. Для вычисления площади треугольника используется формула Герона $S=\sqrt{p(p-x)(p-y)(p-z)}$, где p – полупериметр, вычисляемый по формуле $p=\frac{x+y+z}{2}$, а x, y, z – длины сторон треугольника. Поэтому логично оформить вычисление площади треугольника через подпрограмму с параметрами, а в основной программе сделать обращения к ней.

Использование подпрограммы-процедуры

```
Program SP;
   Var
a,b,c,d,S1,S2,S3,S:real;
(*********************************)
Procedure GERON(x,y,z:real;
                Var SG:real);
Var P:real;
```

Использование подпрограммы-функции

```
Program SF;
 Var a,b,c,d,S:real;
(**********************)
Function
GERON(x,y,z:real):real;
Var P, SG:real;
Begin
```

```
Begin
P:=(x+y+z)/2;
SG:=Sqrt(p*(p-x)*(p-y)*(p-z));
End;
(**********************************)
Begin
  ClrScr;
  Writeln('Ввведите a,b,c,d');
  Readln(a,b,c,d);
  GERON(1,a,2,s1);
  GERON(2,b,2.5,S2);
  GERON(c,2.5,d,S3);
  S:=S1+S2+S3;
  Writeln('Площадь S=',S:6:3);
  Readln;
     End.
```

```
P:=(x+y+z)/2;
SG:=Sqrt(p*(p-x)*(p-y)*(p-z));
GERON:=SG
End;
(**********************************)
Begin
ClrScr;
Writeln('введите a,b,c,d');
Readln(a,b,c,d);
S:=GERON(1,a,2)+GERON(2,b,2.5)
+GERON(c,2.5,d);
Writeln('Площадь  S=',S:6:3);
Readln;
End.
```

Процедура Geron

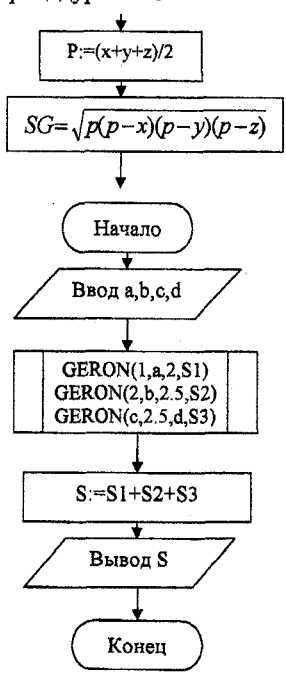


Рис. 9.3. Блок-схема решения задачи с использованием процедуры

Функция Geron

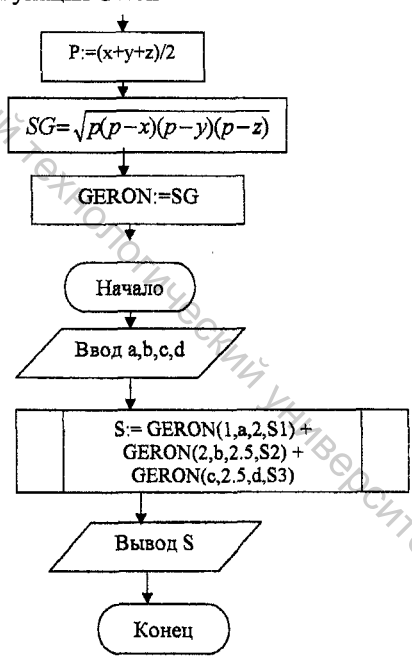


Рис. 9.4. Блок-схема решения задачи с использованием функции

Здесь x, y ,z, SG – формальные параметры. Причем x, y, z – параметры-значения, SG – параметр-переменная. A, b, c, d, S - фактические параметры, P – локальная переменная.

9.6. Рекурсивные подпрограммы

Во многих случаях оптимизация алгоритма решения задачи требует вызова для выполнения подпрограммы из раздела операторов той же самой подпрограммы, т.е. подпрограмма вызывает сама себя. Такой способ вызова называется *рекурсией*. Рекурсия полезна в тех случаях, когда основную задачу можно разделить на подзадачи, имеющие ту же структуру, что и первоначальная задача. Программы, реализующие рекурсию, называются *рекурсивными*.

Прямая рекурсия заключается в том, что подпрограмма в своем блоке использует саму себя.

Пример

```
Program A;
     Procedure B;
          Begin
          Writeln('Пример');
      B;
          End;
     Begin
       B;
     End.
```

Здесь программа A содержит процедуру B, которая будет вызывать сама себя и печатать слово «Пример» бесконечное число раз.

Классическим примером использования рекурсии является вычисление факториала целого положительного числа [3]. Проиллюстрируем на программе.

```
Program DemoRecurs;
 Var N:integer;
 Function Fact(A:integer):integer;
  Begin
  If A=0 Then Fact:=1 Else Fact:=A*Fact(A-1);
  End;
  Begin
   While True do
   Begin
   Writeln('Введите число N');
   readln(N);
   Case N of
   0   : Writeln('0! = 0');
   1..9: Writeln(N,'!=',Fact(N));
   10: Halt
   Else Writeln('Число должно быть в диапазоне 0..9');
```

```
End;
End;
End.
```

Здесь рекурсия используется, чтобы решить подзадачи, затем сложить результаты их решений и получить, таким образом, решения первоначальной задачи.

Часто встречается и другой вариант рекурсии, когда первая подпрограмма вызывает вторую, которая в момент вызова еще не определена. Такая ситуация называется *косвенной рекурсией*. Для реализации косвенной рекурсии используется так называемое предварительное описание процедур и функций.

9.7. Задания для самостоятельной работы

Составить программы на языке Pascal с использованием подпрограмм пользователя. Вид подпрограммы определить самостоятельно.

Вариант 1. Найти расценку на выполнение i-той операции. Вычисление стоимости 1 секунды оформить через подпрограмму. Для определения стоимости 1 секунды по номеру разряда в процедуре использовать оператор CASE.

Расценка = Стоимость 1 секунды * Затрата времени.

Разряд	Стоимость 1 секунды, руб.
2	3,1
3	4
4	4,3

№ операции	Разряд	Затрата времени
1	3	117
2	4	50
3	2	25
4	3	36
5	4	58

Вариант 2. Найти сумму продаж по каждому виду товара. Определение цены по номеру товара оформить через подпрограмму, в которой использовать оператор CASE.

Сумма продаж = цена * количество

№ товара	Цена
101	4600
102	2200
103	3600
104	6800

Месяц	№ товара	Количество
1	102	3
1	101	12
1	103	4
2	101	5
2	104	17
3	104	6
3	103	18
3	102	3

Вариант 3. Найти остаточную стоимость и износ за месяц по каждому виду материальных средств. Вычисление износа за месяц и остаточной стоимости оформить через подпрограмму.

Шифр	Наименование материальных средств	Первоначальная стоимость	Сумма износа	Норма амортизации
П1	Пресс для ВТО	800000	12000	18
Т1	Ткацкий станок Т1	1000000	10000	10
Т2	Ткацкий станок Т2	1500000	15000	11
Ш1	Швейная машина 1022 кл.	300000	5000	16
Ш2	Швейная машина 1097 кл.	350000	7000	15

Износ за месяц вычисляется по формуле $A_{мес} = \frac{C_{пер} * H_a}{12*100}$,

где $A_{мес}$ - сумма износа за месяц, $C_{пер}$ – первоначальная стоимость, H_a - норма амортизации, %.

Остаточная стоимость вычисляется по формуле $O_{ост} = C_{пер} - C_{изн} - A_{мес}$,

где $O_{ост}$ - остаточная стоимость, $C_{изн}$ - сумма износа, $A_{мес}$ - износ за месяц.

Вариант 4. Определить выпуск в смену по моделям, если в потоке обрабатываются 3 модели. Мощность потока (общий выпуск в смену) М = 369 ед. / смена. Соответствие выпуска по моделям: модели А – 2 единицы, модели Б – 1 единица, модели В – 2 единицы.

Выпуск в смену $M_i = \frac{M}{\sum m_i} \cdot m_i$

Расчет выпуска в смену M_i оформить через подпрограмму.

Пояснение: соответствие выпуска по моделям, m_i – отношение количества моделей определенного вида к общему выпуску изделий на потоке.

Например, если обрабатывается 3 модели А, Б, В, их выпуск составляет 180 единиц, 90 единиц и 90 единиц соответственно, а общая мощность потока – 360 единиц в смену, то соотношение выпуска по моделям для модели А равно 2, для модели Б – 1, для модели В – 1, т.е. из 4 единиц 2 изделия относятся к виду А, 1 – к Б, 1 – к В.

Вариант 5. Длины четырех кусков подкладочной ткани артикула С21 составляют 41,9, 80,75, 73,9 и 55,8 м. Определить количество изделий, которое можно раскроить из данных кусков, если норма на обмелку 1 изделия составляет 2,41 м. Определить длину концевого остатка, получаемого в результате раскроя полотна. Определить общее количество изделий и общую длину концевых остатков, образующихся в результате раскроя данных кусков.

Количество изделий = Длина куска / Норма на обмелку.

Полученное значение округлить до ближайшего меньшего целого.

Длина концевого остатка = Длина куска – Количество изделий * Норма на обмелку.

Определение количества изделий, на которое можно раскроить кусок ткани, и длины концевого остатка оформить в подпрограмме.

Вариант 6. Рассчитать основные технико-экономические показатели потока по изготовлению женских блузок 3 моделей А, Б, В, если время выполнения сменного задания для модели А $R_а$=9918,7 с., для модели Б $R_б$=9771,1 с., для модели В $R_в$=9859,7 с., соотношение выпуска m_i по моделям А, Б, В составляет 1:1:1 (т.е. каждой модели последовательно выпускается по 1 единице), трудоемкость изготовления изделий $T_а$=5574 с., $T_б$=5496 с., $T_в$=5544 с.

Продолжительность смены, R – 28800 с., мощность потока в смену, M – 369 единиц. Расчет технико-экономических показателей оформить через подпрограмму.

Показатели:

1. Выпуск каждой модели $M_i = \frac{M}{\Sigma m_i} \cdot m_i$. Полученное значение округлить до целого числа.

2. Такт потока по модели $t_i = \frac{R_i}{M_i}$.

3. Расчетное количество рабочих $N_p = \frac{T_i}{t_i}$

Вариант 7. Определить, используя подпрограмму, годовую экономию текущих затрат по каждому виду работ и экспериментальному цеху в общем для каждого варианта механизации.

Годовая экономия текущих затрат по i-тому виду работ

$$Э'_{год_i} = L_{год_i} \cdot n_i \cdot \frac{k_{a_i}}{100},$$

где $L_{год_i}$ - годовая заработная плата одного работающего по i-тому виду работ, тыс. руб./год, n_i – количество сотрудников, занятых i-тым видом работ, k_{a_i} - ожидаемое снижение трудоемкости по i-тому виду работ, %.

Должность	Количество	Средняя годовая з/п, тыс. руб.	Вариант 1, $Кс_1$	Вариант 2, $Кс_2$	Вариант 3 $Кс_3$
Инженер-расчетчик	6	1650	30%	10%	50%
Конструктор	75	1750	25%	20%	5%
Служащий	18	1200	40%	25%	12%

Общая годовая экономия текущих затрат по экспериментальному цеху

$$Э'_{год} = \sum_{i=1}^{m} Э'_{год_i} \cdot \left(k_{доп} + \frac{k_н}{100}\right),$$

где m – количество видов работ, $k_{доп.}$ - коэффициент, учитывающий дополнительную заработную плату и начисления, $К_{доп}$ – 1,3, $k_н$ – накладные расходы, $К_н$ – 80 %.

Вычисленные значения вывести на экран.

Вариант 8. Составить программу для расчета размера начисленной и расчетной заработной платы бригад и участков ткацкого производства.

Начислено заработной платы = Основная заработная плата + Доплаты за дополнительные наряды + Доплаты за работу во внеурочное время.

Расчетная заработная плата = Начислено заработной платы * КТУ.

Вычисление начисленной и расчетной заработной платы оформить в подпрограмме.

Подразделения	Основная заработная плата, млн. руб.	Доплаты, млн. руб.		КТУ
		За дополнительные наряды	За работу во внеурочное время	
Участок 1	1,5	0,45	0	1,1
Участок 2	2	0,25	0,2	0,9
Участок 3	2,2	0,8	0,4	1,2
Участок 4	3,1	0,21	0,12	1,2
Участок 5	1,8	0,12	0,5	1,3

Вариант 9. Рассчитать эффективность усовершенствования потока по изготовлению женских блузок

Экономия времени, с, определяется по формуле $СЗВ = \frac{T_{ст} - T_{нов}}{T_{ст}} \cdot 100$

Рост производительности труда, %, $РПТ = \frac{T_{ст} - T_{нов}}{T_{нов}} \cdot 100$

Здесь $T_{ст}$ – фабричные затраты времени, $T_{нов}$ – проектируемые затраты времени.

Узел обработки	Затраты времени, с	
	Фабричные	Проектируемые
Обработка полочек	474	240
Обработка воротника	338	254
Обработка рукава	440	452
Обработка кокетки	120	81
Монтаж	2280	2010

Вычисление СЗВ и РПТ оформить через подпрограмму.

Сделать вывод об эффективности усовершенствования потока: если суммарное снижение затрат времени и суммарный рост производительности труда больше нуля, усовершенствования эффективны, в противном случае неэффективны.

Значения СЗВ и РПТ по узлам обработки, а также суммарное снижение затрат времени и суммарный рост производительности труда вывести на экран.

Вариант 10. Определить рациональную мощность потока по изготовлению женских блузок, если такт потока, τ=75 с., затрата времени на единицу, T=5574 с., продолжительность смены, R=28800 с.

Для выбора рациональной мощности рассчитываются следующие технико-экономические показатели при отклонениях от такта потока 80, 90, 100, 105 и 110 %. Расчет показателей оформить с использованием подпрограмм.

1. Расчетный такт потока $\tau_p = \frac{\%откл}{100} \cdot \tau$.

2. Мощность потока в смену, ед., $M = \frac{R}{t_{p_i}}$. Полученное значение округлить до целого.

3. Расчетное количество рабочих $N_p = \frac{T}{t_{p_i}}$.

4. Фактическое количество рабочих Nф принимается округлением расчетного количества до целого.

5. Коэффициент использования рабочего времени $K_{исп} = \frac{N_p}{N_\phi}$.

6. Производительность труда 1 рабочего $П_т = \frac{M}{N_\phi}$.

Мощность рациональна, если $K_{исп}$=1, производительность труда больше средней, а рекомендуемое количество рабочих составляет 50-90 человек.

ГЛАВА 10. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ

10.1. Производные типы (массивы)

До сих пор рассматривались лишь *простые* (а именно – скалярные) типы данных, где каждым значением любого из этих типов является отдельное данное, то есть тривиальная структура.

Значения *производных* типов в общем случае представляют собой нетривиальную структуру, то есть обычно имеют более чем одну компоненту. При этом каждая компонента структуры может быть как отдельным данным, так и нетривиальной структурой, то есть значением любого из производных типов. На языке Pascal можно обрабатывать не только отдельные переменные, но и их совокупности. Одной из таких совокупностей и наиболее часто используемым производным типом является *регулярный* тип, а его значения обычно называют *массивами* [1].

Массив – это структурированный (регулярный) тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип [3].

Тип элементов массива называется *базовым*. Число элементов массива фиксируется при описании и в процессе выполнения программы не меняется. Доступ к каждому отдельному элементу осуществляется путем индексирования элементов массива. Индексы представляют собой выражения любого скалярного типа, кроме вещественного. Тип индекса определяет границы изменения значений индекса.

Например, массив A состоит из элементов a_1, a_2, a_3, . . . a_n.

Здесь a – *имя элемента* массива,

n – его *индекс*, т.е. порядковый номер в массиве, по которому можно определить (найти) каждый элемент.

Для описания массива в языке Turbo Pascal предназначено словосочетание `ARRAY OF` (массив из...). Описание массива возможно двумя способами.

Способ 1. Используя раздел описания типов.

```
Type <имя типа> = array [тип индекса] of <тип компонент>;
     Var <идентификатор>: <имя типа>;
```

Например,
```
Type Mas = array[1..100] of real;
     Var  M1,M2:Mas;
```

Способ 2. Используя раздел описания переменных (Var) без представления типа в разделе описания типов данных:

```
Var
<идентификатор>: array [тип индекса] of <тип компонент>;
```

Например,
```
Var M1, M2: array[1..100] of real;
    V1: array[1..50] of integer;
```

Если в качестве базового типа взят другой массив, образуется структура, которую принято называть *многомерным* массивом.

Например,
```
Type Vector = array [1..4] of integer;
     Massiv = array [1..4] of Vector;
Var  M1:Massiv;
```
Ту же структуру можно получить, используя другую форму записи:
```
Var M1: array[1..4,1..4] of integer;
```
Таким образом, многомерный массив задается при помощи индексированных выражений, следующих друг за другом. Если индексное выражение одно, то массив называется *одномерным*, если два – *двумерным* (матрица), если n – *n-мерным*.

Одномерные массивы в Паскале обычно используются для представления векторов, двухмерные – для представления матриц.

Например,
```
Var Vector = array [1..5] of real; {одномерный массив-вектор из 5-ти элементов типа real}
    Matr = array [1..4,1..3] of integer; {матрица из 4-х строк и 3-х столбцов с элементами типа integer}
```
При описании массивов для задания границ индексов можно использовать предварительно определенные константы. Например,
```
Const G1=4;G2=5;
Var   Matr = array [1..G1,1..G2] of integer;
```
Элементы массива располагаются в памяти последовательно, по строкам. Для массива Matr:

Matr[1,1]	Matr[1,2]	Matr[1,3]
Matr[2,1]	Matr[2,2]	Matr[2,3]
Matr[3,1]	Matr[3,2]	Matr[3,3]
Matr[4,1]	Matr[4,2]	Matr[4,3]

При обращении к элементам массива на Паскале индекс указывается в квадратных скобках: A[i], B[i,j]. Синтаксическая диаграмма типа массив представлена на рис. 10.1.

Рис. 10.1. Синтаксическая диаграмма типа массив

При работе с массивами необходимо помнить следующее:
- массив – это структура данных, представляющая собой совокупность элементов одного типа;
- при объявлении массива указывается количество элементов массива;
- доступ к элементу массива осуществляется путем указания индекса (номера) элемента массива.

Тип индекса элемента массива может быть только скалярным. Наиболее часто в качестве типа индекса используется ограниченный тип [3], причем в большинстве случаев это ограниченный целый тип.

Так, например, массив из 100 компонент вещественного типа может быть задан как `array [1..100] of real;`

Ограниченный целый тип 1..100 определяет количество компонент – 100 и их упорядоченность – от 1-ой до 100-ой.

В большинстве задач нумерация компонент начинается от 1 и ограничивается положительным целым числом. Но это необязательно. Если, например, представить в виде массива численность г. Москвы [3] в отдельные годы, то нумерацию удобно начинать с 1147 – года основания Москвы и до 2006 , т.е. такой массив можно задать как

```
array[1147..2006] of integer;
```

а для Рима, основанного в 754 г. до н.э.

```
array[-754..-1] of integer;
```

здесь –1 –последний год до н.э.

Каждый из рассмотренных массивов в программе может быть задан двояко: 1) используя описание типа:

```
Type Gran = 1..100;
     vector = array[Gran] of real;
     ch_M = array[1147..2006] of integer;
     ch_R = array[-754..-1] of integer;
Var  a, b: vector;
     c: ch_M;
     d: ch_R;
     n: Gran;
```

Обращение к элементам массивов c и d: `c[1147]`, `c[1999]` и т.п., то есть `c[i]`, `d[i]`. Для обращения к элементам массивов a и b нужно ввести переменную типа `Gran`, например `n`. Обращение `a[Gran]` неверно, т.к. `Gran` является типом индекса, а не его именем;

2) непосредственно при описании переменной:

```
Var  a,b: array[1..100] of real;
     c: array[1147..2006] of integer;
     d: array[-754..-1] of integer;
```

Использование *ограниченного целого типа* для индексов дает достаточно широкие возможности для описания массивов.

Множество значений перечисляемого типа также образует *ограниченное перенумерованное множество*. Следовательно, перечисляемый тип также может быть использован в качестве типа индексов [3]. Значения перечисляемого типа упорядочены (порядок задается порядком перечисления имен, являющихся значениями этого типа), и число их конечно (определяется количеством имен в этом типе).

Например, необходимо проанализировать среднемесячную температуру воздуха за год [3]. Можно в качестве индекса использовать ограниченный целый тип 1..12, но можно для наглядности ввести перечисляемый тип, состоящий из имен месяцев года:

```
Type Month = (Jan, Feb, Mar, Apr, … , Dec);
```

Тогда переменные, являющиеся векторами среднемесячных температур, можно определить следующим образом:

```
Var t,r: array[Month] of real;
```

Частичные компоненты массива, обозначающие температуру каждого месяца, запишутся следующим образом:

```
t[Yan], t[Feb], t[Apr] и т. п.
```

Если ввести переменную m типа Month (`m: Month`), то можно обращаться `t[m]`, где m меняется от `Jan` до `Dec`.

В качестве типа индексов могут использоваться также стандартные типы Boolean и Char.

Например

```
Type priznak = array[Boolean] of integer;
     kod_s = array[Char] of integer;
Var k: priznak;
    s: kod_s;
```

Частичные компоненты таких массивов выглядят следующим образом:

```
k[False], k[True];
s['d'], s['h'].
```

Для работы с массивом как с единым целым используется идентификатор массива без указания индекса в квадратных скобках. Если массивы описаны одинаково, то с ними можно проводить следующие операции [1]:

- *Проверка на равенство:* A = B.
  Результатом является значение «истина», если каждый элемент массива A равен соответствующему элементу массива B.
- *Проверка на неравенство:* A <> B.
  Результатом является значение «истина», если значение хотя бы одного элемента массива A не равно значению соответствующего элемента массива B.
- *Присваивание.* A : = B.
  Все значения элементов массива B присваиваются элементам массива A. Значения элементов массива B остаются без изменения.

10.2. Действия над элементами одномерного массива

После объявления массива каждый его элемент можно обработать, указав идентификатор массива и индекс элемента в квадратных скобках. Рассмотрим типичные ситуации, возникающие при работе с массивами. Для этого опишем четыре одномерных массива: два (A и B) из элементов целочисленного типа, два (B и C) из элементов вещественного типа.

```
Var A, B: array[1..5] of integer;
    C, D: array [1..10] of real;
```

Инициализация массива. *Инициализация* – это присваивание каждому элементу массива какого-либо (или одного и того же) значения типа. Например, A[1]:=0; A[2]:=0; ... A[5]:=0. В данном случае индекс изменяется от 1 до 5. Но при большом количестве элементов такой способ инициализации нерационален. Гораздо удобнее получить такой же результат, используя оператор FOR:

```
For i:=1 to 5 do
    A[i]:= 0;
```

Ввод-вывод элементов массива. Pascal не имеет средств ввода-вывода элементов массива сразу, поэтому ввод и вывод значений производится поэлементно. Чаще всего они вводятся с помощью оператора *Read* (*Readln)* с использованием оператора цикла *For*:

```
For i:= 1 to 5 do
  Begin
    Writeln('введите ', i:2, ' элемент массива');
    Readln(B[i]);
  End;
```

Вывод элементов массива производится аналогично, но с использованием оператора `Write (Writeln)`:

```
For i: = 1 to 5 do
Writeln(i:2, ' -й элемент массива=', B[i]:4:1);
```

Копирование элементов из одного массива в другой.

B:=A; (B и A имеют одинаковый тип компонент)

A:=C; (Неверно, т.к. C имеет тип `real`, а A – тип `integer` и массивы A и C разной длины);

Или, используя цикл:

```
For i:= 1 to 5 do
 Begin
    C[i]:= A[i];
    B[i]:= D[i]; неверно, т.к. B - integer; D - real;
```

Поиск в массиве элементов, удовлетворяющих некоторым условиям.

Пусть надо выяснить, сколько элементов в массиве C имеют нулевое значение.

```
K:= 0;
For i:=1 to 10 do
 If C[i] = 0 then k:=k+1;
```

После выполнения цикла переменная K будет содержать количество элементов с нулевым значением.

Вычисление суммы (произведения) элементов массива.

Найти сумму (S) и произведение (P) элементов массива C:

```
Sum:= 0;                          Pr:=1;
For i:= 1 to 10 do                For i: = 1 to 10 do
Sum:= Sum + C[i];                 Pr:= Pr*C[i];
```

Перестановка элементов массива.

Осуществляется с использованием вспомогательной переменной того же типа, что и базовый тип массивов. Например, требуется поменять местами первый и четвёртый элементы массива A:

```
Per:= A[1];
A[1]:= A[4];
A[4]:= Per;
```

Поиск максимального (минимального) элемента в массиве.

```
Max:= A[1];
K:= 1;
  For i:= 2 to 5 do
     If A[i]>Max
          Then Begin Max:= A[i];
                     K:= i;
               End;
```

Для получения более полного представления об основных действиях при обработке массивов разберем примеры.

Пример 10.1.

Вычислить % дней в месяце, когда температура опускалась ниже 0^0C. Для упрощения вычислений ввести следующие обозначения:

`all_d` – всего дней в месяце (примем значение `all_d =31`);

`cold_d` – количество холодных дней;

`temp` – массив температур за месяц;

`pr` – процент дней с отрицательной температурой.

Текст программы `Cold` на языке Pascal приведен ниже.

```
Program Cold;
Uses CRT;
Const all_d =31;              {описание констант и переменных}
Type days = array[1..all_d] of real;
Var temp: days;
    pr: real;
    cold_d: integer;
    i: integer;
Begin
  Writeln('Введите массив температур за месяц -  temp');
     For i:= 1 to 31 do          {ввод значений температуры за
     Read(temp[i]);               каждый из 31 дня месяца}
{подсчет в % количества дней с температурой ниже 0^0}
cold_d:=0;
  For  i := 1 to all_d do
  If temp[i]<0 Then cold_d:=cold_d+1;
    pr:=cold_d/all_d*100;
    Writeln('cold_d=',cold_d:2,' pr=',pr:5:2);
  Delay(10000);
End.
```

Пример 10.2.

Определить средний тарифный разряд рабочих и работ. Сделать вывод, обеспечен ли участок квалифицированными кадрами. Известны: N – число разрядов рабочих, N=5;

M – число разрядов работ, M=3;

R_i – разряды рабочих {2, 3, 4, 5, 6};

P_i – число рабочих по i-му разряду {2, 4, 5, 3, 3};

S_i – разряды работ {2, 4, 5};

T_i – трудоемкость работ по разрядам {120, 180, 210}.

Средний тарифный разряд рабочих определится по формуле

$$RSR = \frac{\sum_{j=1}^{n}(R_j \cdot P_j)}{\sum_{j=1}^{n} P_j} > 1$$

Средний тарифный разряд работ определится по формуле

$$SSR = \frac{\sum_{i=1}^{m}(S_i \cdot T_i)}{\sum_{i=1}^{m} T_i} > 1$$

Если RSR < SSR, то квалификация рабочих не соответствует квалификации работ.

Текст программы `Uchastok` на языке Pascal приведен ниже.

```
Program Uchastok;
 Uses CRT;
 Type mas=array[1..10] of real;
 Var P,R,S,T: mas;
     RSR,SSR: real;
     n,m: integer;
(*****************************************)
 Procedure Vvod(dl:integer;var v:mas);
  Var i:integer;
  Begin
  For i:=1 to dl Do
   Begin
    Writeln('Введите',I:2,'элемент массива');
    Readln(v[i]);
   End;
  End;
(*****************************************)
 Procedure Vyvod(dl:integer; v :mas);
  Var i:integer;
  Begin
  For i:=1 to dl Do
   Begin
    Write(v[i]:3:1,'':2);
```

```
      End;
     End;
   (*****************************************)
     Function SUM(dl:integer;v:mas):real;
      Var i:integer;
        S:real;
      Begin
      S:=0;
      For i:=1 to dl Do
       S:=S+v[i];
      SUM:=S;
      End;
   (*****************************************)
     Function SUM_PR(dl:integer;v,w:mas):real;
      Var i:integer;
        S:real;
      Begin
      S:=0;
      For i:=1 to dl Do
       S:=S+v[i]*w[i];
      SUM_PR:=S;
      End;
   (*****************************************)
     Begin
     ClrScr;
     Writeln('Введите количество разрядов работ - m');
     Readln(m);
     Writeln('Введите количество разрядов рабочих - n');
     Readln(n);
     Writeln('Введите массив разрядов рабочих R');
     Vvod(n,R);
     Writeln('Введите количество рабочих каждого разряда P');
     Vvod(n,P);
     Writeln('Введите массив разрядов работ S');
     Vvod(m,S);
     Writeln('Введите трудоемкость работ по разрядам T');
     Vvod(m,T);
     RSR:=SUM_PR(n,R,P)/SUM(n,P);
     SSR:=SUM_PR(m,S,T)/SUM(m,T);
     ClrScr;
     If RSR>=SSR Then
     Writeln('Участок обеспечен квалифицированными рабочими')
        Else
     Writeln('Участок не обеспечен квалифицированными рабочи-
ми');
     Writeln('Средний разряд рабочих RSR=',RSR:4:1);
     Writeln('Средний разряд работ SSR=',SSR:4:1);
     Writeln('Массив разрядов рабочих');
```

```
Vyvod(n,R);
Writeln;
Writeln('Количество рабочих каждого разряда');
Vyvod(n,P);
Writeln;
Writeln('Массив разрядов работ');
Vyvod(m,S);
Writeln;
Writeln('Трудоемкость работы по каждому разряду');
Vyvod(m,T);
Readln;
End.
```

Результаты

Участок обеспечен квалифицированными рабочими
Средний разряд рабочих RSR=4,1
Средний разряд работ SSR=3,9
Массив разрядов рабочих
2 3 4 5 6
Количество рабочих каждого разряда
2 4 5 3 3
Массив разрядов работ
2 4 5
Трудоемкость работы по каждому разряду
120 180 210

Пояснения к программе.

Для ввода и вывода массивов используются процедуры `Vvod` и `Vyvod`, написанные для формального массива `v` длиной `dl`. Используя эти процедуры, вводятся и выводятся массивы исходных данных (R, P, S, T).

В подпрограмме-функции `SUM` описывается вычисление суммы элементов формального массива `v` длиной `dl`.

В подпрограмме-функции `SUM_PR` описывается вычисление суммы произведений формальных массивов `v` и `w` длиной `dl`.

Далее в программе эти функции используются для вычисления по заданным формулам значений среднего разряда рабочих `RSR` и среднего разряда работ `SSR` для фактических параметров–значений исходных данных.

10.3. Сортировка массивов

Сортировка – это процесс упорядочивания набора данных одного типа по возрастанию или по убыванию значения какого-либо признака.

Массив (например, массив A={a_1, a_2, …, a_{10}}) является отсортированным (упорядоченным) по возрастанию, если для всех i из интервала [1.., n+1] выполняется условие $a_i \le a_{i+1}$.

Различают следующие виды сортировок.

1. Обменные сортировки.

Суть обменных сортировок заключается в следующем. Сначала последовательно просматриваются числа $a_1, \ldots, a_n$ и находится наименьшее i, такое, что $a_i > a_{i+1}$. Затем эти a_i и a_{i+1} меняются местами, и просмотр возобновляется с элемента a_{i+1}, и так далее. Тем самым наибольшее число передвигается на последнее место. Следующие просмотры начинаются опять сначала, уменьшая на единицу количество просматриваемых элементов. Массив будет упорядочен после просмотра, в котором участвовали только первый и второй элементы. В пределах этого вида можно выделить следующие методы сортировок: 1) метод пузырька, 2) метод улучшенной «пузырьковой» сортировки.

Метод «пузырька». Массив последовательно просматривается несколько раз. Каждый просмотр состоит из сравнения каждого элемента массива a_i с элементом a_{i+1} и обмена этих двух элементов, если они расположены не в нужном порядке.

Пример 10.3.

Пусть задан массив А, состоящий из следующих элементов: *25 57 48 37 12 92 86 33.*

На первом просмотре делаем следующие сравнения:

a_1 с a_2 (25 и 57) – нет обмена;

a_2 с a_3 (57 и 48) – обмен;

a_3 с a_4 (57 с 37) – обмен;

a_4 с a_5 (57 с 12) – обмен;

a_5 с a_6 (57 с 92) – нет обмена;

a_6 с a_7 (92 с 86) – обмен;

a_7 с a_8 (92 с 33) – обмен.

Таким образом, после первого просмотра элементы массива будут располагаться в следующем порядке:

25 48 37 12 57 86 33 92.

После первого просмотра наибольший элемент 92 находится в нужной позиции. В общем случае элемент $a_{(n-i+1)}$ будет находиться в нужной позиции после i-той итерации. Этот метод называется *«методом пузырька»* потому, что каждое число медленно всплывает как «пузырек» вверх в нужную позицию.

После второго просмотра получим последовательность

25 37 12 48 57 33 86 92. Элемент *86* на нужном месте.

Поскольку каждая итерация помещает новый элемент в нужную позицию, то для сортировки некоторого массива, состоящего их n элементов, требуется не более *n-1* итерации. Полный набор итераций для массива А выглядит следующим образом:

1-ая итерация 25 48 37 12 57 86 33 92;
2-ая итерация 25 37 12 48 57 33 86 92;
3-ая итерация 25 12 37 48 33 57 86 92;
4-ая итерация 12 25 37 33 48 57 86 92;

5-ая итерация 12 25 33 37 48 57 86 92;
6-ая итерация 12 25 33 37 48 57 86 92;
7-ая итерация 12 25 33 37 48 57 86 92.

Метод улучшенной «пузырьковой» сортировки. Этот метод основан на методе «пузырька» с той лишь разницей, что количество итераций меньше.

Пример 10.4.

Обратимся к условию примера 10.3 и проиллюстрируем метод улучшенной «пузырьковой» сортировки программой Sort1:

```
Program Sort1;
 Uses CRT;
  Var a: array[1..50] of real;
      i,j,m: integer;
          w: real;
 Begin
     Clrscr;
     Writeln('Введите количество элементов массива A');
     Readln(m);
     Writeln('Введите массив A');
     For i:=1 to m do
     Read(a[i]);
     Writeln;
     Writeln('Исходный массив A');
     For i:=1 to m do
       Write(a[i]:4:1,'':2);
     Writeln;
```

{Сортировка массива по возрастанию. При использовании метода улучшенной «пузырьковой» сортировки i изменяется от 1 до m-1, j от 1 до m-i}

```
      For i:=1 to m do
For j:=1 to m-1 do
   If a[j]>a[j+1] then
     begin w:=a[j]; a[j]:=a[j+1]; a[j+1]:=w end;
      Writeln('Отсортированный по возрастанию массив A');
      For i:=1 to m do
        Write(a[i]:4:1,'':2);
        Writeln;
```

{Сортировка массива по убыванию}

```
      For i:=1 to m do
        For j:=1 to m-1 do
          If a[j] < a[j+1] then
          begin  w:=a[j]; a[j]:=a[j+1]; a[j+1]:=w end;
      Writeln('Отсортированный по убыванию массив A');
      For i:=1 to m do
        Write(a[i]:4:1,'':2);
      readkey;
   End.
```

Сеанс работы с программой:
Введите количество элементов массива A
8
Введите массив A
25 57 48 37 12 92 86 33
Результаты:
Исходный массив A
25.0 57.0 48.0 37.0 12.0 92.0 86.0 33.0
Отсортированный по возрастанию массив A
12.0 25.0 33.0 37.0 48.0 57.0 86.0 92.0
Отсортированный по убыванию массив A
92.0 86.0 57.0 48.0 37.0 33.0 25.0 12.0

2. Сортировки выбором. Наиболее распространены такие методы этого вида сортировки, как:

- с поиском минимального значения;
- с использованием бинарных деревьев;
- с использованием метода «турнира с выбыванием»;
- «пирамидальная» сортировка.

Рассмотрим подробнее первый метод этого вида сортировки – *с поиском минимального значения*. Суть его заключается в следующем. Сначала в массиве необходимо найти минимальный элемент, затем поменять его местом с первым, затем в усеченном (исключая первый элемент) массиве опять найти минимальный элемент и поставить его на второе место и так далее.

Пример 10.5.

Отсортировать по возрастанию массив A из элементов 1 3 0 9 2.

Процесс сортировки выбором по методу поиска минимального значения пройдет через следующие шаги:

0:	1	3	0	9	2	min=a[3]=0	переставляем a[1] < -- > a[3]
1:	0 |	3	1	9	2	min=a[3]=1	переставляем a[2] < -- > a[3]
2:	0	1 |	3	9	2	min=a[5]=2	переставляем a[3] < -- > a[5]
3:	0	1	2 |	9	3	min=a[5]=3	переставляем a[4] < -- > a[4]
4:	0	1	2	3	9	Готово	

Здесь знак | отделяет отсортированную часть массива от несортированной. Ниже приводится текст программы, реализующей этот метод.

```
Program sort2;
 Uses CRT;
  Var  a: array[1..50] of real;
       i,j,m,k: integer;
       min,w: real;
 Begin
     Clrscr;
```

```
        Writeln('Введите количество элементов массива A');
        Readln(m);
        Writeln('Введите массив A');
        For i:=1 to m do
        Read(a[i]);
        Writeln;
        Writeln('Исходный массив A');
        For i:=1 to m do
          Write(a[i]:4:1,'':2);
          Writeln;
    {Сортировка массива выбором}
         For i:=1 to m do
         Begin
          min:=a[i]; k:=i;
          For j:=i+1 to m do
            If a[j] <= min then
            Begin min:=a[j]; k:=j end;
    {Перестановка i-го и min элементов}
            w:=a[i];
            a[i]:=a[k];
            a[k]:=w;
         End;
        Writeln('Отсортированный массив A');
        For i:=1 to m do
          Write(a[i]:4:1,'':2);
          Writeln;
          readkey;
    End.
```

Сеанс работы с программой:

Введите количество элементов массива A

5

Введите массив A

1 3 0 9 2

Исходный массив A

1.0 3.0 0.0 9.0 2.0

Отсортированный массив A

0.0 1.0 2.0 3.0 9.0

3. Сортировка «вставками». Наиболее распространены такие методы этого вида сортировки как:

- сортировка «простыми» вставками;
- сортировка «бинарными» вставками;
- сортировка Шелла;
- сортировка с вычислением адреса.

Суть сортировки *«простыми»* вставками заключается в последовательном просмотре массива $a_2, \ldots, a_n$ с тем, чтобы каждый новый элемент a_i

вставлять на подходящее место в уже упорядоченной совокупности a_1, ... , a_{i-1}. Это место определяется последовательным сравнением a_i с упорядоченными элементами a_1, ... , a_{i-1}.

При сортировке «*бинарными*» вставками место, на которое надо вставить элемент a_i в уже упорядоченную совокупности a_1, ... , a_{i-1} определяется алгоритмом деления пополам. Таким образом, получается алгоритм сортировки «*бинарными*» вставками, который понимается как «вставка делением пополам».

4. Сортировка «слияниями» (Алгоритм фон Неймана). Суть заключается в упорядочении массива a_1, ... , a_n по неубыванию. Метод основан на многократных слияниях уже упорядоченных групп элементов массива. Вначале весь массив рассматривается как совокупность упорядоченных групп по одному элементу в каждом. Слиянием соседних групп получаются упорядоченные группы, каждая из которых содержит два элемента (кроме последней группы, которой не нашлось парной). Далее полученные группы укрупняются тем же способом и т.д. Здесь приходится оперировать не только с массивом a_2, ... , a_n , но и с вспомогательным массивом b_1, ... , b_n.

Пример 10.6.

Вставить в упорядоченный по возрастанию массив A = {3 4 7 9} новый элемент, равный 5, таким образом, чтобы сохранилась упорядоченность.

Алгоритм решения задачи следующий:

1. Найти в массиве тот элемент, который больше вставляемого. Это элемент a[3] = 7. Для этого последовательно просмотреть все элементы, начиная с первого.
2. Увеличить длину массива на 1. Получим массив A = {3 4 7 9 X}.
3. Все элементы, стоящие правее от найденного, включая его самого, то есть от 3-го, сдвинуть вправо. Получим массив A = {3 4 7 7 9}.
4. На освободившуюся позицию, то есть на место элемента a[3], вставить новый элемент, то есть 5.

   В итоге получаем массив A = {3 4 5 7 9}.

При решении задач такого рода следует учитывать, что, если все элементы массива меньше вставляемого, новый элемент надо вставить в конец массива, и если все элементы массива больше вставляемого, то новый элемент надо вставить в начало массива.

Ниже приведена программа Sort3, реализующая данный алгоритм:

```
Program Sort3;
 Uses CRT;
  Var a: array[1..50] of real;
      i,j,m,k: integer;
      g:real;
 Begin
     Clrscr;
     Writeln('Введите количество элементов массива A');
     Readln(m);
```

```
    Writeln('Введите массив A');
    For i:=1 to m do
    Read(a[i]);
    Writeln;
    Writeln('Введите число, которое нужно вставить');
    Readln(g);
    Writeln('Исходный массив A');
    For i:=1 to m do
      Write(a[i]:4:1,'':2);
      Writeln;
{1. Ищем элемент больше вставляемого}
k:=1; {k - индекс сравниваемого элемента}
While (k<=m) and (g>=a[k]) do {если k не вышла за границу m и
                    вставляемый элемент меньше или равен a[k]}
        k:=k+1;          {то переходим к следующему элементу}
{2. Увеличиваем длину массива на 1}
   m:=m+1;
{3.Сдвигаем элементы, начиная с k-го вправо}
   For i:=m downto k+1 do
     a[i]:=a[i-1];
{4.В a[k] заносим g}
   a[k]:=g;
   Writeln('Новый массив A');
   For i:=1 to m do
   Write(a[i]:4:1,'':2);
     Writeln;
   readkey;
 End.
```

Сеанс работы с программой:

Введите количество элементов массива A
4
Введите массив A
3 4 7 9
Введите число, которое нужно вставить
5
Исходный массив A
3.0 4.0 7.0 9.0
Новый массив A
3.0 4.0 5.0 7.0 9.0

10.4. Процедуры и функции обработки многомерных массивов

Основные операции по обработке данных, организованных в многомерные массивы, рассмотрим на уровне подпрограмм-процедур и подпрограмм-функций, описанных для формальных параметров. В качестве примера исполь-

зования в составленных ниже программах подпрограмм опишем следующие массивы: *матрицы* A, B, C размера M x N и *одномерные массивы* D и F длиной N ($N \le 10$).

```
Type Matr = Array[1..10,1..10] Of Real;
     Mas = Array[1..10] Of Real;
Var A, B, C: Matr;
     F,D:  Mas;
```

Ввод матрицы по строкам

```
Procedure Vvod(m,n:Integer; Var v:Matr);
   Var i,j:Integer;
    Begin
        For i:=1 To m Do
         Begin
            Writeln('Введите ', i ,' строку матрицы');
              For j:=1 To n Do
                 Readln(v[I,J]);
         End;
    End;
```

Обращение из программы: *Vvod(4,5,A);* или *Vvod(3,5,B);*

Вывод матрицы (в виде матрицы)

```
Procedure Vyvod(M, N:Integer; V:Matr);
   Var i,j:Integer;
    Begin
        For i:=1 To M Do
          Begin
              For j:=1 To N Do
                  Write(V[I,J]:4:1,'':2);
              Writeln;
          End;
    End;
```

Обращение из программы: *Vyvod(4,5,A);* или *Vyvod(3,5,B);*

Вычисление суммы элементов матрицы

```
Function Sum(M,N:Integer;V:Matr):Real;
Var i,j:Integer;
          S:Real;
    Begin
       S:=0;
       For I:=1 To M Do
         For J:=1 To N Do
         S:=S+V[I,J];
             Sum:=S;
    End;
```

Обращение из программы: *Sa:=Sum(4,5,A);* или *Sb:=Sum(3,5,B);*

Суммирование диагональных элементов матрицы

```
Function Sled(M:Integer;V:Matr):Real;
Var i:Integer; S:Real;
     Begin
        S:=0;
        For i:=1 To M Do
          S:=S+V[i,i];
               Sled:=S;
     End;
```

Обращение из программы: *Sla:=Sled(4,A); или Slb:=Sled(3,B);*

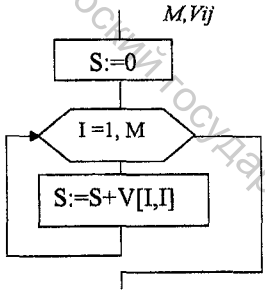


Рис.10.2. Блок-схема вычисления следа матрицы

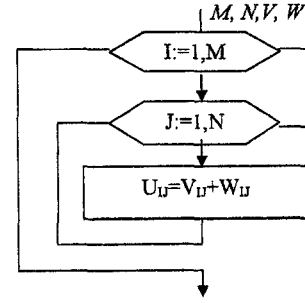


Рис.10.3. Блок-схема вычисления суммы матриц

Суммирование матриц

```
Procedure Sum_Matr(M,N:Integer;V,W:Matr;Var U:Matr);
Var i,j:Integer;
        Begin
          For i:=1 To M Do
For j:=1 To N Do
          U[i,j]:=V[i,j]+W[i,j];
        End;
```

Обращение из программы: *Sum_Matr(3,5,A,B,D);*

Суммирование матрицы по строкам

```
Procedure Sum_Str(M,N:Integer;V:Matr;Var S:Mas);
     Var i,j:Integer;
        Begin
         For i:=1 To M Do
         Begin
          S[i]:=0;
          For j:=1 To N Do
          S[i]:=S[i]+V[i,j]
         End
        End;
```

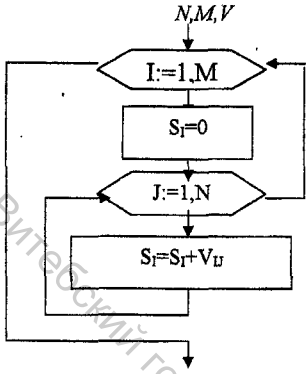


Рис.10.4. Блок-схема вычисления суммы строк матрицы

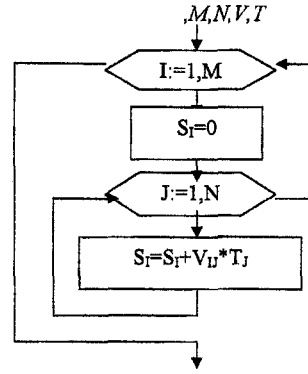


Рис. 10.5 Блок-схема умножения матрицы на вектор

Обращение из программы:
Sum_Str(4,5,A,S_A);
Sum_Str(3,5,B,S_B);

Умножение матрицы на вектор

$Fi = \sum_{J=1}^{M} A_{IJ} * D_J$. Длина исходного вектора равна количеству столбцов матрицы. Длина результирующего вектора равна количеству строк матрицы.

```
Procedure Vector(M,N:Integer;V:Matr;T:Mas;Var S:Mas);
    Var i,j:Integer;
       Begin
         For i:=1 To M Do
        Begin
         S[i]:=0;
         For j:=1 To N Do
S[i]:=S[i]+V[i,j]*T[j];
        End;
       End;
```

Обращение из программы:
Vector (4,5,A,D,F);

Удаление строки матрицы

Требуется удалить строку с заданным номером К. Все строки, начиная с (К+1)-ой, нужно переместить вверх. Число строк уменьшится на 1.

```
Procedure Udstr(M,N,Kk:Integer;V:Matr;Var W:Matr);
Var i,j:Integer;
   Begin
     W:=V;
```

```
 M:= M-1;
For i:=Kk To M Do
 For j:=1 To N Do
 W[i,j] := W[i+1,j];
End;
```

Обращение из программы:

Udstr (4,5,K,A,D);

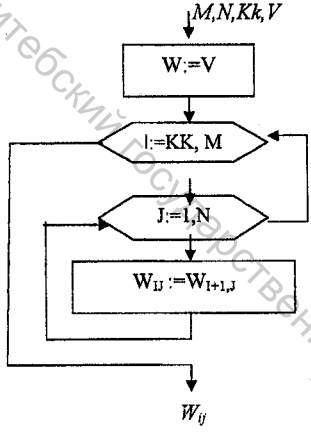


Рис.10.6. Блок-схема удаления строки матрицы

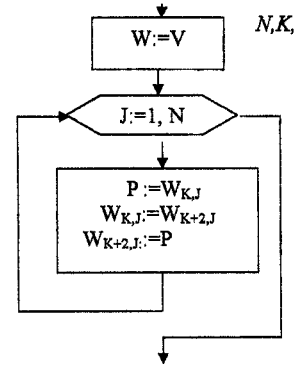


Рис. 10.7. Блок-схема перестановки строк матрицы

Перестановка строк матрицы

Пусть требуется поменять местами К-ую и (К+2)-ую строки.

Рассмотрим два варианта алгоритма:

а) с использованием вспомогательной переменной Р.

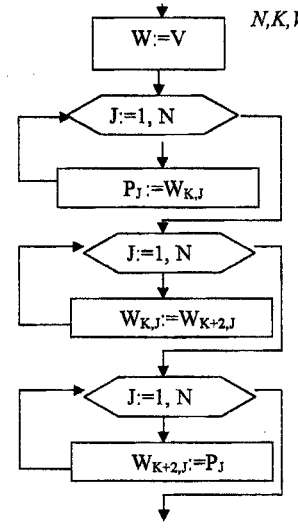
```
Procedure Perest(N,K:Integer;V:Matr;Var W:Matr);
    Var i,j:Integer;
          P:Real;
       Begin
         W:=V;
         For j:=1 To N Do
        Begin
          P:=W[K,J];
          W[k,j] := W[i+2,j];
          W[k+2,j]:=P;
         End;
       End;
```

Обращение из программы:

Perest (5,K,A,D);

б) с использованием вспомогательного массива P_j: одна из строк пересылается целиком в массив P_j для временного хранения:

```
Procedure Perest1(N,K:Integer;
     V:Matr; Var      W:Matr);
     Var i,j:Integer;
          P:Mas;
       Begin
         W:=V;
         For j:=1 To N Do
         P[j]:=W[k,j];
              For j:=1 To N Do
         W[k,j]:= W[k+2,j];
         For j:=1 To N Do
         W[k+2,j]:=P[j];
        End;
```

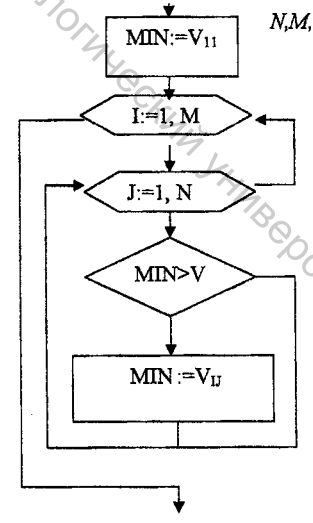


Обращение из программы:
Perest1 (5,2 ,A,D);

Рис. 10.8. Блок-схема перестановки строк матрицы

Поиск минимального (максимального) элемента матрицы

```
Procedure Min_El(M,N:Integer;
V:Matr; Var Min:Real;
Var L,K:Integer);
     Var i,j:Integer;
     Begin
      Min:=V[1,1];
      K:=1; L:=1;
      For i:=1 To M Do
           For j:=1 To N Do
If Min>V[i,j] Then
           Begin
              Min:=V[i,j];
         K:=i;
         L:=j;
              End;
       End;
```



Обращение к процедуре:
Min_El(4,5,A,Min_A,K_A,L_A);

Рис.10.9. Блок-схема поиска минимального элемента матрицы

Поиск минимального элемента строки (столбца) матрицы

```
Procedure Min_Str(M,N:Integer;V:Matr;Var Min:Mas);
Var i,j:Integer;
Begin
 For i:=1 To M Do
     Begin
    Min[i]:=V[I,1];
          For j:=2 To N Do
If Min[i]>V[i,j] Then
           Min[i]:=V[i,j];
          End;
    End;
```

Обращение к процедуре:
Min_El(4,5,A,Min_A);

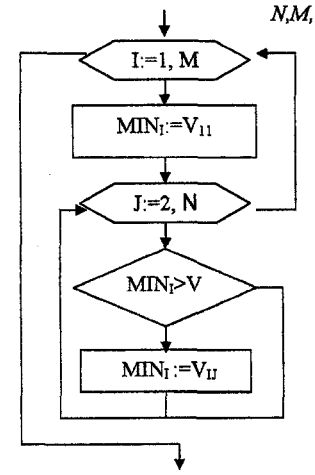


Рис 10.10. Блок-схема поиска максимального элемента строки матрицы

10.5. Использование массивов при решении экономических задач

Пример 10.7.

Установить соответствие производственной программы выпуска продукции производственной мощности цеха.

В механосборочном цехе завода четыре производственных участка, на которых установлено следующее технологическое оборудование (N_i): токарные станки – 17 шт., фрезерные – 11 шт., протяжные – 14 шт., шлифовальные – 26 шт. Эффективный фонд времени – **F**=3931 час в год. Среднее выполнение норм выработки =110% (**K**=1,1).

Заданная программа характеризуется данными таблицы 4.5:

Таблица 10.1. Исходные данные

	Обозначение	А	Б	В
Количество комплектов на программу выпуска	(Q_i)	300	700	500
Плановая трудоемкость токарных работ	$T_{пкij}$	75	35	50
Плановая трудоемкость фрезерных работ		35	25	38
Плановая трудоемкость протяжных работ		40	32	34
Плановая трудоемкость шлифовальных работ		70	80	60

Сначала определяется станкоемкость планируемой продукции по видам работ

$$ST_i = \frac{\sum_{j=1}^{m} Q_j * T_{mxtj}}{K}, \text{ ч/год}$$

где m – число комплектов оборудования (в данном случае – 3 – А, Б, В);

К – коэффициент выполнения норм выработки.

Далее определяется действительный фонд времени по группам оборудования

$$FD_i = F \cdot N_i.$$

Затем определяется коэффициент загрузки оборудования

$$KZ_i = \frac{ST_i}{FD_i} \le 1.$$

Таким образом, исходными данными к задаче являются одномерные массивы N = {17, 11, 14, 26} и Q = {300, 700, 500}, а также матрица трудоемкостей по видам работ $T = \begin{vmatrix} 75 & 35 & 50 \\ 35 & 25 & 38 \\ 40 & 32 & 34 \\ 70 & 80 & 60 \end{vmatrix}$.

Ниже приводится программа решения этой задачи на языке Pascal.

```
Program Zagruzka;
 Uses Crt;
 Type matr=array[1..4,1..3] of integer;
      mas=array[1..4] of real;
     Var T:matr;
         N,Q,S,ST,FD,KZ:mas;
         F,i,j:integer;
         k:real;
(************************************)
 Procedure Vvod(m:integer;var v:mas);
   Begin
    For i:=1 to m do Readln(v[i]);  End;
(************************************)
   Procedure Vvod1(m,l:integer;var v:matr);
   Begin
    For i:=1 to m do
    Begin  Writeln('введите ',i:2,' строку матрицы');
     For j:=1 to l do readln(v[i,j]); End;
   End;
(************************************)
 Procedure Vyvod(m:integer; v:mas);
  Begin
   For i:=1 to m do
    Write(v[i]:5:2,'':2);
    Writeln;
   End;
 (************************************)
  Procedure Vyvod1(m,l:integer; v:matr);
```

```
Begin
 For i:=1 to m do
 Begin
  For j:=1 to l do
  Write(v[i,j]:3);
  Writeln;
 End;
End;
(************************************)
Begin
Clrscr;
Writeln('Введите количество оборудования каждого вида N');
Vvod(4,N);
Writeln('Количество комплектов на программу выпуска Q?');
Vvod(3,Q);
Writeln('Трудоемкость по видам работ и оборудования T?');
Vvod1(4,3,T);
Writeln('Введите эффективный фонд времени F');
Readln(F);
Writeln('Введите коэффициент k');
Readln(k);
For i:=1 to 4 do
 Begin
  S[i]:=0;
  For j:=1 to 3 do
   S[i]:=Q[j]*T[i,j];
  ST[i]:=S[i]/k;
 End;
For i:=1 to 4 do
  Fd[i]:=F*N[i];
For i:= 1 to 4 do
  KZ[i]:=ST[i]/FD[i];
Writeln('Количество комплектов на программу выпуска Q');
Vyvod(3,Q);
Writeln('Количество оборудования каждого вида N');
Vyvod(4,N);
Writeln('Трудоемкость по видам работ и оборудования T');
Vyvod1(4,3,T);
Writeln('Станкоемкость продукции ST');
Vyvod(4,ST);
Writeln('Действительный фонд времени FD');
Vyvod(4,FD);
Writeln('Коэффициенты загрузки оборудования KZ');
Vyvod(4,KZ);
Readln;
End.
```

10.6. Задания для самостоятельной работы

Вариант 1. Сведения о количестве проданных товаров каждым продавцом представлены в виде целочисленного вектора из 20 элементов. Найти количество продавцов, объем продаж у которых больше нуля. Перед вводом объема продаж каждого продавца должна выводиться подсказка с номером продавца (1, 2, 3 и т.д.).

Вывести на экран исходный массив и количество продавцов.

Данные для отладки:

Pr = {20; 0; 30; 23; 34; 0; 10; 9; 27; 45; 0; 24; 35; 17; 29; 24; 16; 34; 0; 8}.

Вариант 2. Сведения о количестве работников каждого подразделения предприятия, имеющих надбавку за профессионализм, представлены в виде вектора из 20 элементов. Найти среднее арифметическое число работников, имеющих надбавку. Учесть, что в некоторых подразделениях таких работников нет.

Вывести на экран исходный массив и среднее количество работников, имеющих надбавку.

Данные для отладки:

K = {2; 0; 3; 2; 1; 0; 1; 0; 2; 4; 0; 2; 3; 1; 2; 2; 1; 3; 0; 2}.

Вариант 3. Выпуск продукции ткацкими цехами предприятия представлен в виде вектора из 10 элементов. Найти, сколько цехов предприятия выпустили продукции в объеме от 300 до 500 погонных метров.

Вывести на экран исходный массив и количество цехов.

Данные для отладки:

Vypusk = {600; 500; 300; 390; 640; 500; 500; 490; 270; 450).

Вариант 4. Индексы потребительских цен представлены в виде вектора из 15 элементов. Найти, сколько раз встречается минимальный и максимальный индекс. Переменной К присвоить значение 1, если количество максимальных элементов больше количества минимальных, -1, если количество минимальных больше количества максимальных, и 0, если значения равны.

Вывести на экран исходный массив, сколько раз встречаются минимальный и максимальный индексы, значение переменной К.

Данные для отладки:

Index = {344; 139,3; 163,1; 281,7; 344; 207,5; 146,1; 134,8; 125,4; 114,4; 108; 106; 110; 125,4; 106).

Вариант 5. Объемы производства химических волокон представлены в виде вектора из 15 элементов. Найти количество периодов, в которых выпуск продукции превышал средний за все периоды.

Вывести на экран исходный массив, количество периодов, средний выпуск продукции.

Данные для отладки:

Vypusk = {600; 500; 300; 390; 640; 500; 500; 490; 270; 450; 340; 420; 380; 450; 500).

Вариант 6. Сведения о выпуске трикотажных изделий, млн. шт., предприятиями Республики Беларусь представлены в виде вектора из 15 элементов. Найти средний арифметический выпуск трикотажных изделий, среднее квадратическое отклонение и коэффициент вариации этого показателя.

Вывести на экран исходный массив, средний арифметический выпуск, среднее квадратическое отклонение и коэффициент вариации данного показателя.

Среднее квадратическое отклонение рассчитывается по формуле

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

где n – количество элементов вектора;

x_i – значение элемента вектора;

$\bar{x}$ - среднее арифметическое значение элементов.

Коэффициент вариации, %, вычисляется по формуле

$$CV = \frac{\sigma}{\bar{x}} \cdot 100.$$

Данные для отладки:

Vypusk = {39; 37; 45; 59; 63; 59; 49; 38; 41; 39; 41; 45; 42; 49; 50).

Вариант 7. Количество мушек в 10 опытных партиях чесальной ленты из химических волокон представлено в виде вектора. Определить, является ли опыт b браком.

Мушка – закатанные в комочек волокна, не разделяющиеся без обрыва.

Определение брака в опытах проводится по критерию Стьюдента

$$\eta = \frac{y - \bar{y}}{\sigma},$$

где y – значение переменной, вызывающей сомнение;

$\bar{y}$ - среднее значение переменной, определяется при исключении сомнительного опыта из начального расчета, т.е. при n-1;

σ - среднее квадратическое отклонение, $\sigma = (y_{max} - y_{min}) \cdot K$, где K – коэффициент, зависящий от числа испытаний. При n=9 K=0,31.

Если величина $\eta \geq t = 2$, то опыт является браком.

Вывести на экран исходный массив, среднее значение $\bar{y}$, среднее квадратическое отклонение σ, минимальное и максимальное количество мушек и вывод, является ли опыт b браком.

Данные для отладки:

Mushki = {6; 15; 5; 7; 9; 8; 10; 5; 6; 10), опыт №2.

Вариант 8. Сведения о производстве тканей, млн. м^2, предприятиями Республики Беларусь, представлены в виде вектора из 10 элементов. Найти коэффициент асимметрии. Сделать вывод о смещении кривой распределения в правую или левую сторону.

Вывести на экран исходный массив, значения $\bar{x}$, x' и x'', коэффициент асимметрии, а также вывод о смещении кривой распределения.

Коэффициент асимметрии

$$A = \sqrt{\frac{\pi}{2}} \frac{x' + x'' - 2\bar{x}}{x'' - x'},$$

где $\bar{x}$ - среднее арифметическое;

x' - среднее из тех значений, которые меньше $\bar{x}$;

x'' - среднее из тех значений, которые больше $\bar{x}$

Если A=0, то кривая симметрична, A<0 – кривая смещена в правую сторону, A>0 – кривая смещена в левую сторону.

Данные для отладки:

Tkani = {239; 259; 299; 265; 287; 262; 254; 267; 305; 293).

Вариант 9. Выпуск обуви, млн. пар, представлен в виде вектора из 15 элементов. Найти суммарный и средний выпуск обуви без учета минимального и максимального объемов выпуска.

Вывести на экран исходный массив, значения минимального, максимального объемов, суммарного и среднего выпусков обуви без учета минимума и максимума.

Данные для отладки:

Vypusk = {13; 11,4; 15,6; 16,2; 16,5; 15,4; 13,8; 11,7; 10,5; 10,7; 10,1; 11; 12,8; 12,5; 13).

Вариант 10. Масса волокна, выброшенная из весового питателя за 10 последовательных бросков, представлена в виде вектора. Определить потери сырья, %, в процессе тонкогребенного чесания без останова машины.

$$П = \left(1 - \frac{M}{G}\right) \cdot 100,$$

где G – суммарная масса волокна за 10 последовательных бросков весового питателя, г;

M – масса ленты, наработанной машиной за время t, равное продолжительности поступления 10 бросков, г.

При t=3,12 мин. M=2880 г.

Вывести на экран исходный массив, суммарную массу волокна, потери сырья.

Данные для отладки:

Massa = {310; 305; 298; 308; 294; 312; 306; 319; 219; 301).

Вариант 11. Известны межотраслевые потоки продукции 4 отраслей народного хозяйства, представленные в таблице 10.2. Найти валовый выпуск продукции $X_{отч}$ по каждой отрасли.

Таблица 10.2. Межотраслевые потоки продукции

Отрасль	I	II	III	IV	$Y_{пл}$
I	40	65	23	48	116
II	24	50	26	40	110
II	18	36	42	27	70
IV	34	47	38	45	90

Валовый выпуск продукции $X_{отч}$ представляется собой сумму межотраслевых потоков и конечной продукции отрасли $Y_{пл}$ и рассчитывается как

$$x_i = \sum_{j=1}^{n} x_{ij} + y_i ,$$

где x_{ij} - межотраслевые потоки продукции;

i и j - соответственно номера отраслей производящих и потребляющих;

y_i - конечная продукция i-той отрасли, $i = 1,\overline{n}$, $j = 1,\overline{n}$.

Межотраслевые потоки продукции, вектор конечной продукции и вектор валового выпуска продукции вывести на экран.

Вариант 12. Известны межотраслевые потоки продукции 4 отраслей народного хозяйства, представленные в таблице 10.3, и валовый выпуск продукции $X_{отч}$ = {292, 250, 193, 254}. Составить матрицу коэффициентов прямых затрат на выпуск продукции по каждой отрасли.

Таблица10.3. Межотраслевые потоки продукции

Отрасль	I	II	III	IV
I	40	65	23	48
II	24	50	26	40
II	18	36	42	27
IV	34	47	38	45

Коэффициент прямых затрат a_{ij} показывает, какое количество продукции i-той отрасли необходимо, учитывая только прямые затраты, для производства единицы продукции j-той отрасли.

Коэффициент прямых затрат является довольно стабильной величиной во времени и вычисляется по формуле

$$a_{ij} = \frac{x_{ij}}{x_i}, \; i = 1,\overline{n}, \; j = 1,\overline{n},$$

где x_{ij} - межотраслевые потоки продукции;

i и j - соответственно номера отраслей производящих и потребляющих;

x_i - валовый выпуск продукции i-той отраслью.

Матрицу коэффициентов прямых затрат, межотраслевые потоки продукции и вектор валового выпуска продукции вывести на экран.

Вариант 13. В таблице приведены сведения о массе клочков, мг, компонентов двух видов (Канекарон UHD и Канекарон RMK) в чесальной ленте для искусственного меха. Найти массу каждого клочка пробы и определить процентное содержание каждого компонента в смеси для каждого клочка пробы.

Канекарон UHD	Канекарон RMK
509	326
224	524
222	452
405	515
215	381
320	461
318	675
271	550
111	419
239	390

Масса клочка пробы определяется как сумма масс клочков канекарона RMK и канекарона UHD. Процентное содержание компонента вычисляется по формуле

% содержание = Масса клочка компонента/ Масса клочка пробы * 100.

Исходную матрицу, вектор значений масс клочков проб, матрицу процентного содержания компонентов в смеси вывести на экран. При решении задачи возможно использование подпрограмм.

Вариант 14. Определить среднюю длину резки волокна канекарон RMK, $l_{ср}$, мм, по формуле

$$l_{ср} = \frac{\sum(l_i n_i)}{\sum n_i},$$

где l_i – длина определенной группы волокон, мм;

n_i – количество волокон длиной l_i, шт.

Длины резки и количество волокон представлены в таблице.

Исходную матрицу и значение средней длины резки волокна вывести на экран.

Длина резки l_i	Количество волокон, n_i шт.
40	2
42	23
44	72
46	3
48	0

Вариант 15. В магазине установлены скидки за покупку товаров: 5% - за покупку товаров на сумму, превышающую 20000 рублей и 3% - за покупку товаров в ночное время. В таблице представлены сведения о сумме покупок, сумме покупок со скидками 5 % и 3 %. Сформировать вектора пятипроцентной скидки и скидки 3% за покупку товаров в ночное время. В каждом векторе найти минимальную и максимальную сумму скидки.

Сумма покупки	Со скидкой 5%	Со скидкой 3% за покупки в ночное время
20 150	19 143	19 143
8 750	8 750	8 750
22 600	21 470	21 470
21 360	20 292	20 292
5 250	5 250	5 093
12 896	12 896	12 509
24 070	22 867	22 181

Исходную матрицу, вектора скидок, минимальные и максимальные значения скидок вывести на экран. При решении задачи возможно использование подпрограмм.

Скидка = Сумма покупки – Сумма покупки со скидкой (3% или 5 %).

Вариант 16. В таблице представлены установленные значения параметров заправки кипоразрыхлителя. Определить минимальное и максимальное значение каждого параметра.

Скорость питающей решетки, v, м/мин	0,23	0,24	0,27	0,37	0,27	0,21
Скорость подающей решетки, v, м/мин	0,37	0,37	0,56	0,62	0,38	0,32
Скорость наклонной игольчатой решетки, v, м/мин	35,1	34,1	34,6	35,1	35,1	35,1

Исходную матрицу и вектора максимальных и минимальных значений параметров вывести на экран. При решении задачи возможно использование подпрограмм.

Вариант 17. В таблице представлены сведения о массе бросков, г, весового питателя чесальной машины. Определить среднее значение массы броска и коэффициент вариации броска по массе для каждой чаши.

Коэффициент вариации, %, вычисляется по формуле

$$CV_j = \frac{\sigma_j}{\bar{x}_j} \cdot 100,$$

где σ_j - среднее квадратическое отклонение, рассчитывается по формуле

$$\sigma_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

где n – количество элементов опытов (10);

x_i – значение элемента;

$\bar{x}_j$ - среднее арифметическое значение элементов.

Левая чаша	Правая чаша
250	220
190	220
200	220
355	230
300	245
235	220
350	210
365	225
190	215
250	220

Исходную матрицу, вектора средних значений масс бросков и коэффициентов вариации по массе вывести на экран.

Вариант 18. В таблице представлено количество часов по декадам, отработанное каждым рабочим раскройного цеха.

Определить минимальное и максимальное отработанное время рабочими раскройного цеха, а также номера декад.

ФИО рабочего	Отработанное время по декадам		
	I	II	III
Иванов А.А.	60	62	60
Буров М.А	58	58	66
Яшин Г.Р.	60	65	65
Козлов Е.Л.	40	40	60
Яковлев П.Л.	50	50	70
Гранов П.П.	65	65	68
Петров С.И.	20	70	70

Исходную матрицу, значения минимального и максимального отработанного времени и номера декад вывести на экран.

Вариант 19. В таблице представлены сведения о количестве мушек в 10 пробах чесальной ленты для искусственного меха «под норку», причем в 1 столбце приведено количество крупных мушек, во 2 – средних, в 3 – мелких. Средняя масса пробы равна 3,4 г.

Определить качество прочеса, если допустимое число средних мушек в 1 г для чесальной ленты из химического волокна – 11, в противном случае прочес считается некачественным.

Мушка – закатанные в комочек волокна, не разделяющиеся без обрыва.

Для решения задачи перевести крупные и мелкие мушки в средние с учетом того, что мелкая мушка оценивается в 1/3 средней мушки, крупная — в 3 средние мушки и найти среднее количество мушек.

Исходную матрицу, количество средних мушек с учетом перевода крупных и мелких, среднее количество мушек в 1 г прочеса вывести на экран.

0	8	10
2	8	10
0	5	3
0	3	2
1	4	7
1	10	3
4	10	5
3	7	11
1	4	12
2	8	10

Вариант 20. В таблице приведены сведения о массе клочков, мг, компонентов трех видов (лавсан, нитрон Н-4, нитрон Н-2)в чесальной ленте для искусственного меха. Найти среднюю массу клочка пробы каждого компонента.

Исходную матрицу и вектор средней массы клочков вывести на экран.

Лавсан	Нитрон Н-4	Нитрон Н-2
635	592	348
650	204	611
554	238	507
561	473	513
850	370	226
540	309	409
587	181	351
687	406	497
678	163	515
656	294	422

ТЕМА 4

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКИХ ЗАДАЧ

ГЛАВА 11. ПРИБЛИЖЕННОЕ РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

В практических вычислениях довольно часто приходится решать уравнения вида:

$$f(x)=0, \qquad (11.1)$$

где функция *f(x)* определена и непрерывна на некотором конечном или бесконечном интервале $a \le x \le b$.

Если функция представляет собой многочлен, то уравнение $f(x)=0$ называют алгебраическим, если же в функцию f(x) входят элементарные (тригонометрические, логарифмические, показательные и т.п.) функции, то такое уравнение называется трансцендентным.

Всякое значение x*, обращающее функцию f(x) в ноль, т.е. такое, что

$$f(x^*)=0,$$

называют *корнем уравнения*, а способ нахождения этого значения x* и есть *решение уравнения*.

Найти корни уравнения вида $f(x)=0$ точно удается лишь в частных случаях. Кроме того, часто уравнение содержит коэффициенты, известные лишь приблизительно, и, следовательно, сама задача о точном определении корней уравнения теряет смысл. Поэтому разработаны методы численного решения уравнений, которые позволяют отыскать приближенные значения корней этого уравнения.

При этом приходится решать две задачи:

1. Отделение корней, т.е. отыскание достаточно малых областей, в каждой из которых заключен только один корень уравнения.
2. Вычисление корней с заданной точностью.

При выделении областей, в которых находятся действительные корни уравнения, можно воспользоваться тем, что если на концах некоторого отрезка непрерывная функция *f(x)* принимает значения разных знаков, то на этом отрезке уравнение $f(x)=0$ имеет хотя бы один корень. Однако выполнение этого условия не гарантирует, что на данном интервале содержится только один корень.

Для выделения областей, содержащих один и только один корень, можно воспользоваться графическим способом. При нахождении интервала изоляции корня средствами языка Паскаль рекомендуется использовать табулирование функции на заданном интервале, т.е. получение значений функции y=f(x) на интервале $[x_{min}; x_{max}]$ при дискретном изменении x, равном dx.

Для решения второй задачи существуют многочисленные приближенные методы, из которых рассмотрим *метод простой итерации, метод Ньютона и метод половинного деления.*

Полагаем, что нам известен отрезок $a \le x \le b$, внутри которого существует и располагается один и только один корень уравнения $f(x) = 0$.

Задача состоит в нахождении этого корня.

11.1. Метод простой итерации

Уравнение $f(x) = 0$ представим в виде

$$x = \varphi(x), \tag{11.2}$$

что всегда можно сделать и притом многими способами. Например, можно выделить из уравнения (11.1) переменную x, остальное перенести в правую часть (это и будет $\varphi(x)$). Или умножить левую и правую часть уравнения на произвольную константу λ и прибавить к левой и правой частям x, т.е. представить (11.1) в виде

$$x = x + \lambda f(x).$$

При этом $\varphi(x) = x + \lambda f(x)$.

Выберем на отрезке [α, β] произвольную точку x_0 - нулевое приближение и примем в качестве следующего приближения:

$$x_1 = \varphi(x_0),$$
$$x_2 = \varphi(x_1),$$

и вообще пусть x_n получается из x_{n-1} по формуле

$$x_n = \varphi(x_{n-1}). \tag{11.3}$$

Этот процесс последовательного вычисления чисел x_n *(n=1, 2, 3,...)* по формуле (10.3) называется *методом итераций.*

Если на отрезке [α; β], содержащем корень x=x* уравнения (11.2), а также его последовательные приближения $x_0, x_1, \ldots .x_n, \ldots$, вычисляемые по методу итераций, выполнено условие

$$|\varphi'(x)| \le q < 1, \tag{11.4}$$

то процесс итераций сходится, т.е., увеличивая n, можно получить приближенное, сколь угодно мало отличающееся от истинного значения корня x*.

Процесс итераций следует продолжать до тех пор, пока для двух последовательных приближений x_{n-1} и x_n не будет обеспечено выполнение неравенства

$$|x_n - x_{n-1}| \le \frac{1-q}{q}\varepsilon; \qquad (11.5)$$

при этом всегда будет выполнено неравенство

$$|x^* - x_n| \le \varepsilon,$$

где ε – заданная абсолютная точность корня x*.

Или в более простом виде $|x^* - x_n| \le \varepsilon, -x_n| \le \varepsilon,$

$$|x_n - x_{n-1}| \le \varepsilon, \qquad (11.6)$$

при выполнении которого также будет обеспечена заданная точность определения корня x*.

При практическом нахождении корней по методу итераций нужно при переходе от уравнения (11.1) к уравнению (11.2) стремиться представить $\varphi(x)$ так, чтобы производная $\varphi'(x)$ по абсолютной величине была возможно меньше 1. В таком случае корень всегда будет найден, и чем меньше величина q, тем меньшее число итераций для этого потребуется.

Пример 11.1

Методом итераций найти корень уравнения

$$f(x) = \arcsin(2x+1) - x^2 = 0, \qquad (11.7)$$

расположенный на отрезке [-0.5; 0] с абсолютной погрешностью $\varepsilon = 10^{-4}$. Определить число итераций, необходимое для нахождения корня.

Уравнение (11.7) преобразуем к виду (11.2) следующим образом:

$$\arcsin(2x+1) = x^2 \Rightarrow \sin(\arcsin(2x+1)) = \sin(x^2) \Rightarrow 2x+1 = \sin(x^2)$$

Это уравнение легко может быть преобразовано к виду $x = \varphi(x)$, где

$$\varphi(x) = 0{,}5(\sin x^2 - 1). \qquad (11.8)$$

Находим $\varphi'(x) = x \cdot \cos x^2$.

Очевидно, $|\varphi'(x)| = |x \cos x^2| \le 0{,}5$ для всех $-0{,}5 \le x \le 0$. Поэтому $q=0{,}5$ и процесс итераций сходится.

За начальное приближение можно принять любую точку отрезка [-0,5; 0], например, $x_0 = -0{,}4$.

Алгоритм нахождения корня уравнения (11.8) представляет следующую последовательность действий:

1. Полагаем x = -0,4, ε = 0,0001 и n = 0.
2. Вычисляем следующее приближение x_{n+1} по формуле
$$x_{n+1} = 0{,}5(\sin x_n^2 - 1) \qquad (11.9)$$
3. Вычисляем разность $\delta = x_{n+1} - x_n$ и увеличиваем величину n на единицу.

4. Проверяем условие $|\delta| > \varepsilon$. Если это условие выполняется, то возвращаемся к вычислению следующего приближения x_{n+1} по формуле (11.9), т.е. к п.2. Если условие не выполняется, т.е. $|\delta| \le \varepsilon$, то результатом считаем величину x_{n+1} и заканчиваем вычисления. При этом число n будет равно числу выполненных итераций.

При составлении блок-схемы и программы вводить переменную с индексом x_n нет необходимости, поскольку результатом будет одно число (корень уравнения), а все последовательные приближения в конечном счете не нужны. На каждом этапе вычислений необходимо помнить лишь два соседних приближения, поэтому приближение x_n обозначим через x, а приближение x_{n+1} – через y. По окончании каждой итерации будем полагать x = y.

Таким образом, метод итераций для решения уравнений может быть представлен в виде блок-схемы, представленной на рис. 11.1.

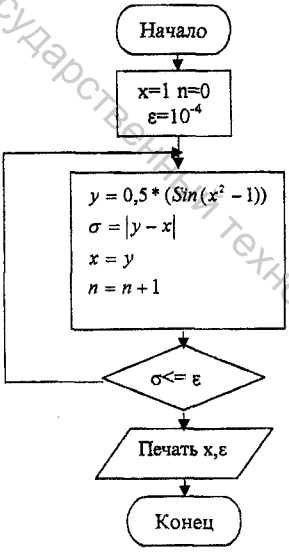


Рис. 11.1. Блок-схема решения уравнения (11.7) методом итераций

Программную реализацию метода итераций представим в двух вариантах:

Вариант 1: с использованием подпрограммы-функции для описания решаемого уравнения, приведенного к виду x=φ(x):

```
Program Primer;
Var a,x,y,e,delta:real;
    n:integer;
```

```
(*************************************)
Function F:real;
Begin                                    { подпрограмма-функция,
 y:=0.5*(sin(x*x)-1);                      описывающая уравнение
 F:=y;                                     y= 0,5(sin(x)^2 – 1)}
End;
(*************************************)
Begin
Writeln('Введите a, e?');
Readln(a,e);
n:=0;
x:=a;
Repeat
n:=n+1;
y:=F;
delta:=abs(y-x);
x:=y;
Until delta<=e;
Writeln('x=',x:7:5,' n=',n:2);
End.
```

Вариант 2: с использованием подпрограммы-функции для описания решаемого уравнения и подпрограммы процедуры для описания метода итераций.

```
Program Primer_A;
Var a,x,y,e,delta:real;
    n:integer;
(*************************************)
Function F:real;
Begin                                    {подпрограмма-функция,
 y:=0.5*(sin(x*x)-1);                     описывающая уравнение
 F:=y;                                    y= 0,5(sin(x)^2 – 1)}
End;
(*************************************)
Procedure Iter;                          {подпрограмма-процедура,
 Begin                                   описывающая метод
  n:=0;                                  итераций}
     x:=a;
  Repeat
   n:=n+1;
   y:=f;
   delta:=abs(y-x);
   x:=y;
  Until delta<=e;
 End;
(*************************************)
Begin                                    {основная программа}
 Writeln('Введите a, e?');
```

```
 Readln(a,e);
 Iter;
 Writeln('x=',x:7:5,'  n=',n:2);
End.
```

В обоих случаях получен *результат x=-0.41453 n=7.*

11.2. Метод Ньютона

Пусть уравнение $f(x)=0$ имеет один корень на отрезке [α; β], причем $f'(x)$ и $f''(x)$ определены, непрерывны и сохраняют постоянные знаки на отрезке [α; β].

Рассмотрение метода Ньютона начнем с его графического представления (рис. 11.1).

Возьмем некоторую точку x_0 отрезка [α; β] и проведем в точке $P_0(x_0, f(x_0))$ графика функции касательную к кривой $y=f(x)$ до пересечения с осью *Ox*. Абсциссу x_1 точки пересечения можно взять в качестве приближенного значения корня. Проведя касательную через новую точку $P_1(x_1, f(x_1))$ и найдя точку ее пересечения с осью *Ox*, получим второе приближение корня x_2. Аналогично определяются следующие приближения.

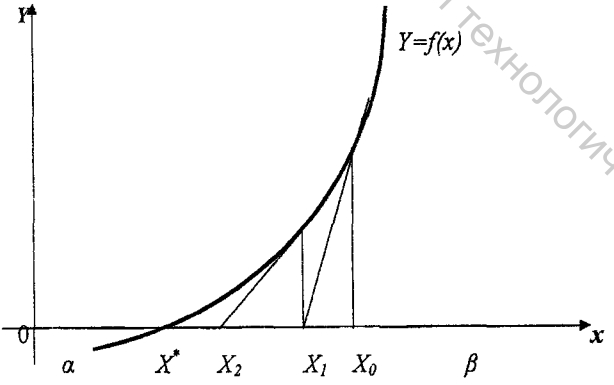


Рис 11.2. Графическое представление метода Ньютона

Выведем формулу для последовательных приближений к корню. Уравнение касательной, проходящей через точку P_0, имеет вид:

$$y = f(x_0) + f'(x_0)(x - x_0).$$

Полагая y=0, находим абсциссу x_1 точки пересечения касательной с осью Ox:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Следующие приближения находим соответственно по формулам

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

...

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}.$$

Процесс вычисления приближений прекратим при выполнении условия:

$$|x_n - x_{n-1}| \le \varepsilon,$$

где ε – заданная абсолютная погрешность корня x^*.

Начальное приближение корня x_0 следует выбирать так, чтобы было выполнено условие

$$f(x_0)f''(x_0) > 0. \qquad (11.10)$$

В противном случае сходимость метода Ньютона не гарантируется.

Чаще всего выбирают $x_0=\alpha$ или $x_0=\beta$, в зависимости от того, для какой из этих точек выполняется условие (11.10).

Замечание

Метод Ньютона эффективен для решения тех уравнений, для которых значение модуля производной $f'(x)$ близ корня достаточно велико, т.е. график функции f(x) в окрестности данного корня имеет большую крутизну.

Пример 11.2

Методом Ньютона найти корень уравнения

$$f(x) = \sin x - x + 0{,}15 = 0, \qquad (11.11)$$

расположенный на отрезке [0,5;1] с абсолютной погрешностью $\varepsilon=10^{-4}$. Нахождение значений y=f(x) оформить через подпрограмму функции.

Решение

Находим $f'(x) = \cos x - 1$. Расчетная формула для уравнения по методу Ньютона имеет вид:

$$x_n = x_{n-1} - \frac{\sin x_{n-1} - x_{n-1} + 0{,}15}{\cos x_{n-1} - 1},$$

где n=1, 2, 3, ...

Начальное приближение x_0 выбираем таким образом, чтобы выполнялось условие $f(x_0)f''(x_0) > 0$. Так как $f''(x) = -\sin x < 0$ для всех $0{,}5 \le x \le 1$, то необходимо подобрать x_0, для которого $f(x_0)<0$. Очевидно, что это условие выполняется при $x_0=1$. Это значение и выберем за начальное приближение корня.

Алгоритм нахождения корня представляет собой следующую последовательность действий:

1. Полагаем $x_0 = 1$, $\varepsilon = 10^{-4}$, $n = 0$.

2. Вычисляем следующее приближение

$$x_{n+1} = x_n - \frac{\sin x_n - x_n + 0{,}15}{\cos x_n - 1}.$$

3. Вычисляем разность $\delta = x_{n+1} - x_n$ и увеличиваем n на единицу.
4. Проверяем условие $|\delta| \le \varepsilon$. Если это условие не выполняется, то возвращаемся к вычислению следующего приближения корня (п.2).Если это условие выполняется, то за результат принимаем величину x_{n+1} и заканчиваем вычисления. При этом число n равно числу выполненных итераций. Как и в методе итераций, переменную с индексом x_{n+1} не вводим, приближение x_n обозначим через x, x_{n+1} – через y.

Таким образом, блок-схема, реализующая метод Ньютона для решения алгебраических и трансцендентных уравнений, может быть представлена в следующем виде (рис. 11.3):

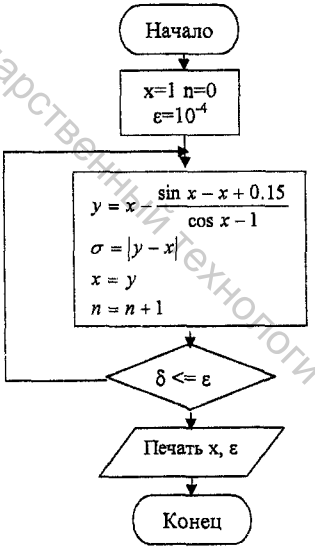


Рис.11.3 Блок-схема решения уравнения (11.11) по методу Ньютона

Программу для решения данного уравнения, как и в предыдущем случае, составим с использованием подпрограммы-функции для описания уравнения и подпрограммы-процедуры для реализации метода решения:

```
Program met_new;
Uses CRT;
 Var x,y,a,eps,delta:real;
     n:integer;
(************************************)
 Function F:real;
```

```
  Begin
   F:=(sin(x)-x+0.15)/(cos(x)-1);
  End;
(************************************)
Procedure Metod;
Begin
 n:=0;
 Repeat
  y:=x-f;
  delta:=abs(y-x);
  x:=y;
  n:=n+1;
 Until delta<eps;
End;
(************************************)
Begin
 Writeln('Введите начальное приближение корня a');
 Readln(a);
 Writeln('Введите точность eps');
 Readln(eps);
 x:=a;
 Metod;
 Writeln('x=', x:5:3, ' n=',n:2);
 Readln;
End.
```

Результат: *x=0.98112 n=2*

11. 3. Метод половинного деления

Пусть дано уравнение *f(x)=0*, где функция *f(x)* непрерывна на отрезке $[\alpha,\beta]$ и $f(\alpha)*f(\beta)<0$. Если вид функции *f(x)* достаточно сложный, то вычисление функций $f'(x)$ в методе Ньютона и $\varphi'(x)$, необходимой для оценки сходимости в методе итераций, затруднительно. Для решения таких уравнений можно использовать метод половинного деления, который хотя и требует значительного объема вычислительной работs, но всегда приводит к результату.

Для нахождения корня уравнения *f(x)=0*, принадлежащего отрезку $[\alpha,\beta]$, делим отрезок пополам, т.е. выбираем начальное приближение равным $x_0=(\alpha+\beta)/2$. Если $f(x_0)=0$, то x_0 является корнем уравнения. Если $f(x_0)\neq 0$, то выбирают тот из отрезков $[\alpha,x_0]$ или $[x_0,\beta]$, на концах которого функция имеет противоположные знаки. Полученный отрезок снова делим пополам и проводим то же рассмотрение и т.д.

Процесс деления отрезков пополам продолжают до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданной точности вычислений ε.

Пример 11.3

Методом половинного деления найти корень уравнения

$$f(x)=x-\sqrt{9+x}+x^2-4=0 \qquad (11.12)$$

на отрезке [2,3] с абсолютной погрешностью $\varepsilon=10^{-4}$.

Алгоритм нахождения корня уравнения (11.12) представляет следующую последовательность действий:

1. Полагаем $\alpha=2$, $\beta=3$ и $\varepsilon=0{,}0001$.
2. Вычисляем $f(\alpha)=\alpha-\sqrt{9+\alpha}+\alpha^2-4=0$
3. Вычисляем $x=(\alpha+\beta)/2$ и значение функции $f(x)=x-\sqrt{9+x}+x^2-4=0$ в этой точке.
4. Проверяем условие $f(x)=0$. Если это условие выполняется, то считаем x корнем и заканчиваем вычисления. Если условие не выполняется, то переходим к выбору отрезка, на концах которого функция f(x) имеет разные знаки.
5. Проверяем условие $f(\alpha)*f(x)<0$. Если это условие выполняется, то полагаем $\beta=x$ и переходим к пункту 6. Если условие не выполняется, то полагаем $\alpha=x$, $f(\alpha)=f(x)$ и переходим к пункту 6.
6. Проверяем условие $\beta-\alpha<=\varepsilon$. Если оно не выполняется, то возвращаемся к пункту 3. Если условие не выполняется, то за результат принимаем значение x и заканчиваем вычисления.

Ниже представлены блок-схема программы решения уравнения (11.12) методом половинного деления (рис. 11.4) и программа на языке Pascal.

```
Program Met_PD;
 Var x,y,z,eps,a,b:real;
     n:integer;
(*********************************)
 Function F(xx:real):real;
  Begin
  f:=xx-sqrt(9+xx)+sqr(xx)-4
  End;
(*********************************)
 Procedure Metod;
 Begin
  y:=f(a); n:=0;
   Repeat
    x:=(a+b)/2;  z:=f(x);
     If z<>0 Then if z*y>0 Then
     Begin a:=x;  y:=z End
        Else b:=x;
     n:=n+1;
   Until (b-a)<eps;
 End;
```

```
(*******************************)
 Begin
 Writeln('Введите a,b');
 Readln(a,b);
 Writeln('Введите eps');
 Readln(eps);
 Metod;
 Writeln('x=',x:5:3,'  n=',n:2);
 Readln;
 End.
```

Результат: *x=2.25775*

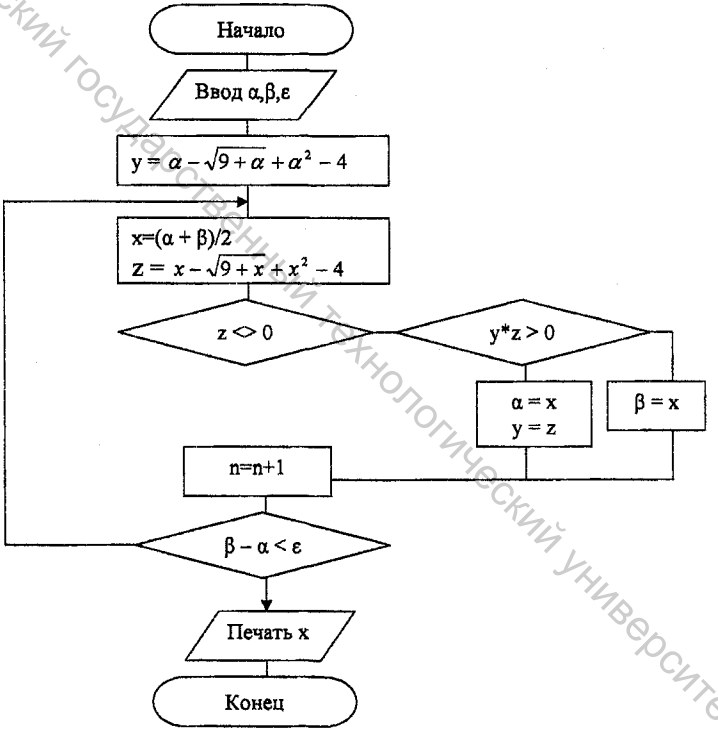


Рис. 11.4. Блок-схема решения уравнения методом половинного деления

11.4. Задания для самостоятельной работы

Выделить интервал изоляции корня уравнения и вычислить корень указанным методом с точностью $\varepsilon = 10^{-4}$.

Вариант	Уравнение	Метод решения
1.	$2\sin\sqrt{x} + 0{,}28x - 1{,}2 = 0$	Метод Ньютона
2.	$2e^{-x} - 2e^{x} - 1 = 0$	Метод Ньютона
3.	$x - 2\sqrt{x} + 3\cdot\sqrt[3]{x} - 4 = 0$	Метод простой итерации
4.	$2{,}7 + \sin\sqrt{x} - x = 0$	Метод простой итерации
5.	$\cos x - \frac{1}{2}\cos^2 x + \frac{1}{4}\cos^4 x - \frac{1}{8} = 0$	Метод половинного деления
6.	$2 - x + \cos x - \ln(x+3) = 0$	Метод простой итерации
7.	$e^{x} + \ln 2x - 4{,}2x = 0$	Метод Ньютона
8.	$\sin\frac{2}{x} + \frac{1}{3}\cos\frac{4}{x} - \frac{3}{x} + \frac{1}{5} = 0$	Метод половинного деления
9.	$2x + 14 - e^{x} + e^{-x} = 0$	Метод Ньютона
10.	$0{,}5x - \frac{2}{5 + \cos(3{,}2x)} = 0$	Метод простой итерации
11.	$2\ln x^2 + 4{,}3\ln x - 2{,}9 = 0$	Метод Ньютона
12.	$x + \sin\left(x^{0{,}31} + 5{,}4\right) = 0$	Метод простой итерации
13.	$\sin^2\frac{x}{2} + \cos^2\frac{x}{4} - 6x = 0$	Метод половинного деления
14.	$x - \sin\left(1 - 0{,}3x^3\right) = 0$	Метод простой итерации
15.	$2x\ln x - \sin x + \cos x = 0$	Метод Ньютона
16.	$0{,}4\cdot 2^{x} - 6{,}7x + 3 = 0$	Метод половинного деления
17.	$\ln x + x - 3{,}4 = 0$	Метод простой итерации
18.	$0{,}3x^2 - 2{,}1x\cdot\ln x = 0$	Метод Ньютона
19.	$\sqrt{2-x} - \sqrt[4]{2+x} + 1 = 0$	Метод половинного деления
20.	$2{,}8x - 1{,}3\ln x - e^{x} = 0$	Метод Ньютона

ГЛАВА 12. ЧИСЛЕННЫЕ МЕТОДЫ ИНТЕГРИРОВАНИЯ

Довольно часто в научно-технических задачах возникает необходимость вычисления определенного интеграла или значений первообразной функции [10]. Приведем некоторые определения.

1. Функция *F(x)* в данном интервале называется *первообразной функцией* для функции *f(x)*, если во всем этом интервале *f(x)* является производной для функции *F(x)*, т. е.

$$F'(x) = f(x).$$

2. Пусть функция *f(x)* задана на некотором отрезке [a; b]. Разобьем этот отрезок произвольным образом на части $\Delta x_i = x_{i+1} - x_i$ (*i=0, 1, 2, ..., n-1*). Возьмем в каждой из частей произвольную точку ξI (*i=0, 1, ..., n-1*). Тогда *определенным интегралом* $\int_a^b f(x)dx$ функции f(x) на отрезке от a до b называется

$$\lim_{\max_i \Delta x_i \to 0} \sum_{i=0}^{n-1} f(\xi_i)\Delta x_i .$$

Условия существования первообразной и определенного интеграла функции *f(x)* рассматриваются в курсе математического анализа. В этом же курсе доказываются предложения:

а) если *F(x)* есть первообразная функция для *f(x)* в интервале, включающем [*a, b*], то $\int_a^b f(x)dx = F(b) - F(a)$;

б) если существует $\int_a^b f(x)dx$, то одной из первообразных функций на отрезке [*a, b*] для *f(x)* является $\int_a^x f(t)dt$.

Таким образом, вычисляя первообразную функцию, можно вычислять определенный интеграл и наоборот. Но, как правило, выразить первообразную функцию через элементарные функции не удается. Поэтому приходится прибегать к приближенному интегрированию. Для решения этой задачи существуют различные численные методы, из которых мы рассмотрим два: *метод трапеций* и *метод Симпсона*.

12.1. Метод трапеций

Как известно, величина определенного интеграла $\int_a^b f(x)dx$ представляет собой площадь криволинейной трапеции, ограниченной графиком функции *f(x)*, осью абсцисс и двумя прямыми *x=a* и *x=b*.

Разобьем отрезок интегрирования $[a;\ b]$ на n равных частей длины $h=\frac{b-a}{n}$. В точках разбиения $x_0=a$, $x_1=a+h, \ldots$, $x_n=b$ проведем ординаты $y_0, y_1, \ldots, y_n$ до пересечения с кривой $y=f(x)$, т.е. $y_i=f(x_i)$, $x_i=a+ih$, $i=0, 1, \ldots, n$. Концы ординат соединим прямолинейными отрезками. Тогда площадь криволинейной трапеции $aABb$ приближенно можно считать равной площади фигуры, ограниченной ломаной линией $aACD \ldots NBb$. Площадь этой фигуры, которую мы обозначим через S, равна сумме площадей трапеций

$$S=h\left(\frac{y_0-y_1}{2}+\frac{y_1+y_2}{2}+\ldots+\frac{y_{n-1}+y_n}{2}\right)=\frac{b-a}{2n}\left(y_0+2y_1+2y_2+\ldots+2y_{n-1}+y_n\right).$$

Таким образом, приближенное значение интеграла по формуле трапеций запишется в виде

$$\int_a^b f(x)dx\approx\frac{b-a}{2n}\left(y_0+2y_1+\ldots+2y_{n-1}+2y_n\right) \qquad (12.1)$$

При составлении программ целесообразно формулу трапеций (12.1) представить в виде

$$\int_a^b f(x)dx\approx\frac{b-a}{2n}\left(y_0+y_n+\sum_{i=1}^{n-1}2y_i\right).$$

Пример 12.1

Найти интеграл $\int_0^{\frac{\pi}{2}}\cos x dx$ методом трапеций, разбивая отрезок интегрирования на n=30 частей.

Для вычисления определенного интеграла используются величины

$h_1=\frac{\pi/2-0}{30}=60$ — шаг интегрирования;

$x_i=ih_1$, $y_i=\cos x_i$, $i=0, 1, .., 30$.

При разработке программы используем две подпрограммы-функции: в первой опишем интегрируемую функцию, во второй – метод трапеций. При составлении блок-схемы и программы нет необходимости в использовании индексированной переменной x_i, так как при вычислении очередного элемента суммы требуется лишь одно значение x, которое может быть получено прибавлением шага h к предыдущему значению x.

Далее приводится программа и результат ее выполнения.

```
Program Int_TR;
Uses crt;
Var a,b,integr,x:real;
            n:integer;
```

```
(*********************************)
Function Fun(z:real):real;
 Begin
  Fun:=cos(z)
 End;
(*********************************)
Function Integral(a1,b1:real;n1:integer):real;
 Var i:integer;
     h1,x,s:real;
 Begin
  h1:=(b1-a1)/n1;
  s:=0;
  For i:=1 to n1-1 do
   Begin
    x:=a1+i*h1;
    s:=s+2*fun(x);
   End;
  Integral:=h1/2*(Fun(a1)+Fun(b1)+s);
 End;
(*********************************)
 Begin
  Writeln('Введите a,b,n');
  Readln(a,b,n);
  Integr:=Integral(a,b,n);
  Writeln('Интеграл=',integr:7:4);
  Readln;
 End.
```

Результат:

Итнеграл = 0,9998

12.2.Метод Симпсона

Разобьем отрезок интегрирования $[a;\ b]$ на $2n$ равных частей длины $h=\frac{b-a}{2n}$. Пусть точкам разбиения $x_0=a$, $x_1=a+h, \ldots$, $x_n=b$ соответствуют значения подынтегральной функции $y_0, y_1, \ldots, y_{2n}$, т. е. $y_i=f(x_i)$, $x_i=a+ih$, $i=0, 1, \ldots, 2n$. На отрезке $[x_0;x_2]$ подынтегральную функцию $f(x)$ заменим параболой, проходящей через точки (x_0,y_0), (x_1,y_1), (x_2,y_2). Уравнение параболы запишем в виде интерполяционной формулы Ньютона

$$f(x)=f(x_0)+\tilde{x}\Delta f(x_0)+\frac{\tilde{x}(\tilde{x}-1)}{2}\Delta^2 f(x_0)=y_0+\tilde{x}\Delta y_0+\frac{\tilde{x}(\tilde{x}-1)}{2}\Delta^2 y_0,$$

где

$$\tilde{x}=\frac{x-x_0}{h};\Delta y_0=y_1-y_0;$$

$$\Delta^2 y_0=\Delta y_1-\Delta y_0=y_2-2y_1+y_0.$$

Интегрируя полученное выражение, имеем (учитывая, что y_0, y_1, y_2 не зависят от x)

$$\int_{x_0}^{x_2} f(x)dx = \int_{x_0}^{x_2} \left\{ y_0 + (y_1 - y_0) + (y_2 - 2y_1 + y_0)\frac{\tilde{x}(\tilde{x}-1)}{2} \right\} dx = \frac{h}{3}(y_0 + 4y_1 + y_2).$$

Аналогично

$$\int_{x_2}^{x_4} f(x)dx = \frac{h}{3}(y_2 + 4y_3 + y_4),$$

.....

$$\int_{x_{2n-1}}^{x_{4n}} f(x)dx = \frac{h}{3}(y_{2n-2} + 4y_{2n-1} + y_{2n})$$

Сложив эти равенства, получаем формулу

$$\int_a^b f(x)dx \approx \frac{b-a}{6n}(y_0 + 4y_1 + 2y_2 + \ldots + 4y_{2n-1} + y_{2n}), \tag{12.2}$$

известную под названием *формулы Симпсона.*

При составлении программ целесообразно формулу Симпсона (12.2) представить в виде

$$\int_a^b f(x)dx = \frac{h}{3}\left(y_0 + y_{2n} + \sum_{i=1}^{2n-1}(3 + c_i)y_i \right),$$

где

$$c_i = \begin{cases} +1, \text{при } i \text{ нечетном} \\ -1, \text{при } i \text{ четном} \end{cases}$$

Пример 12.2.

Вычислить интеграл $\int_0^{\pi/2} \sqrt{1 - \frac{1}{4}\sin^2 x}\,dx$ методом Симпсона, разбивая отрезок интегрирования на $2n$=20 частей.

Для вычисления определенного интеграла имеем

$h_1 = \frac{\pi/2 - 0}{2n} = \frac{\pi}{40}$,

$x_i = ih_1$,

$y_i = \sqrt{1 - \frac{1}{4}\sin^2 x_i}$, i=0, 1, 2, ..., 20.

По соображениям, изложенным в предыдущем примере, нет необходимости в использовании индексированных переменных c_i и x_i. Значения c и x, используемые при вычислении очередного члена, могут быть получены из предыдущих значений c и x рекуррентно, а именно $c=-c$, $x=a1+i\cdot h$.

Далее приводится программа и результат ее выполнения.

```
Program Int_Simp;
Uses crt;
Var a,b,integral,x:real;
    n:integer;
(*********************************)
Function Fun(z:real):real;
Begin
 Fun:=sqrt(1-1/4*sqr(sin(z)))
End;
(*********************************)
Function Simpson(a1,b1:real;n1:integer):real;
 Var i,c:integer;
     h1,x,s:real;
 Begin
  h1:=(b1-a1)/n1;  s:=0; c:=-1;
  For i:=1 to n1-1 do
   Begin
    x:=a1+i*h1;  c:=-c;
    s:=s+(3+c)*fun(x);
   End;
  Simpson:=h1/3*(fun(a1)+fun(b1)+s);
 End;
(*********************************)
 Begin
  Writeln('Введите a,b,n');
  Readln(a,b,n);
  integal:=Simpson(a,b,n);
  Writeln('Интеграл=',integral:7:4);
  Readln;
 End.
```

Результат:

Интеграл=1,4674

12.3. Задания для самостоятельной работы

Найти значение определенного интеграла. Метод вычисления и количество отрезков задать самостоятельно.

Вариант	Условие	Вариант	Условие
1	$\int_1^4 \frac{\ln^2 x}{x} dx$	11	$\int_0^1 \frac{x \cdot arctgx}{\sqrt{1+x^2}} dx$
2	$\int_2^3 \frac{1}{x \lg x} dx$	12	$\int_0^1 \frac{1}{(3\sin x + 2\cos x)^2} dx$
3	$\int_1^2 \frac{x^3}{3+x} dx$	13	$\int_{\pi/6}^{\pi/3} tg^2 x + ctg^2 x dx$
4	$\int_0^1 xe^x \sin x dx$	14	$\int_0^{3/4} \frac{1}{(x+1)\sqrt{x^2+1}} dx$
5	$\int_1^{2.5} \frac{1}{x^2} \sin \frac{1}{x} dx$	15	$\int_0^{1.5} \frac{e^x(1+\sin x)}{1+\cos x} dx$
6	$\int_0^2 \frac{1}{\sqrt{9+x^2}} dx$	16	$\int_0^1 \frac{x^2-1}{(x^2+1)\sqrt{x^4+1}} dx$
7	$\int_1^2 \frac{\sqrt{x^2-0.16}}{x} dx$	17	$\int_1^3 x^x(1+\ln x) dx$
8	$\int_1^{1.5} \sin x \ln(tg(x)) dx$	18	$\int_0^{1.9999} x^2\sqrt{4-x^2} dx$
9	$\int_1^3 x^x(1+\ln x) dx$	19	$\int_0^3 \arcsin\sqrt{\frac{x}{1+x}} dx$
10	$\int_0^{\ln 2} \sqrt{e^x - 1} dx$	20	$\int_1^2 \left(\frac{\ln x}{x}\right)^3 dx$

ГЛАВА 13. СИСТЕМА КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAPLE

13.1. Общие сведения. Правила записи выражений

Maple – система компьютерной математики, позволяющая решать сложные математические задачи без дополнительного программирования. Подробнее об этом см. в [6].

Работа Maple организована в диалоговом режиме: вопрос – ответ в отдельном блоке. Блок выделяется слева квадратной скобкой, длина которой зависит от размеров и количества исходных выражений (вопросов) и результатов вычислений (ответов). Строка ввода математических выражений имеет отличительный символ >.

Алфавит языка содержит 26 прописных и строчных латинских букв (от A до Z и от a до z), 10 арабских цифр (0 – 9) и 32 специальных символа. Идентификатор должен быть уникальным, начинаться с буквы и может содержать буквы, цифры и знак подчеркивания. Maple различает прописные и строчные символы.

Выражение задается с помощью функций и операторов, записываемых в командной строке. Результат вычислений (по умолчанию) возвращается в виде математических формул. Ввод выражения завершается символом фиксации конца выражения – *точкой с запятой*, если ответ выводится в ячейку вывода, или *двоеточием*, если ответ не выводится.

Выражения формируются из операторов и операндов. *Операндами* могут быть константы, переменные и значения функций.

В Maple могут использоваться следующие операторы:

+ - оператор сложения	-> - функциональный оператор
- - оператор вычитания	< - менее чем
* - умножение	> - более чем
/ - деление	= - равно
**, ^ - возведение в степень	<= - менее чем и равно
! – факториал	>= - более чем и равно
. – десятичная точка	<> - неравно
:= - оператор присваивания	or – логическое ИЛИ
and – логическое И	

Целые числа в Maple записывают цифрами в десятичной записи. При записи рациональных чисел для выделения числителя и знаменателя используют оператор деления, причем рациональные числа автоматически упрощаются. Кроме того, они могут быть представлены в десятичном виде и в форме вещественного числа с плавающей точкой с использованием степени 10.

Пример

```
> 0.0036e-10;
```

$$0.36\ 10^{-12}$$

Десятичная точка служит указанием необходимости получить численное значение результата. Для целочисленного аргумента выражение не вычисляется.

Переменные получают значения в результате операции присваивания. Maple помнит последнее присвоенное значение для любой переменной. Для отмены операции присваивания используется функция `evaln(x)` или выражение `x:='x'`. Для отмены всех присваиваний служит команда `restart`.

Выражение можно задавать, используя встроенные функции или создавая новые. Функция в выражениях вводится указанием ее имени и списка из одного или ряда фактических параметров в круглых скобках.

Пример

```
> fun1:=x^2+y^2;# функция задана как выражение
```

$$fun1 := x^2 + y^2$$

```
> fun2:=(x,y)->x^2+y^2;# функция задана с использованием
функционального оператора ->
```

$$fun2 := (x, y) \to x^2 + y^2$$

```
> fun2(2,5);# вызов функции с параметрами 2, 5
```

$$29$$

Большинство команд Maple записываются в символах нижнего регистра.

Результат вычислений может быть получен в символьном виде, в виде численного выражения или числа.

Для простейших вычислений используется функция `evalf(expr, n)` – вычисляет `expr` и возвращает вычисленное значение в форме с плавающей точкой, имеющей `n` значащих цифр. Параметр `n` является необязательным, при его отсутствии `n=10`.

Пример

```
> restart;
> evalf(Pi,3);
```

$$3.14$$

Условные выражения

Для подготовки разветвляющихся программ в Maple-язык включен оператор условия if.

Формат

```
if <Условие сравнения> then <Элементы>
{elif <Условие сравнения> then <Элементы>}
{else <Элементы>}
fi;
```

В { } указаны необязательные элементы.

Чаще используются следующие конструкции:

- `if <Условие сравнения> then <Элементы1> fi`

если `<Условие сравнения>` выполняется, то исполняются `Элементы 1`, иначе ничего не выполняется.

- `if <Условие сравнения> then <Элементы1> else <Элементы2> fi`

если `<Условие сравнения>` выполняется, то исполняются `Элементы1`, иначе исполняются `Элементы2`.

В условиях используются любые логические конструкции со знаками сравнения (>,<,>=,<=,<>,=) и логические операторы `and`, `or`, `not`, конструкции с которыми возвращают значения `true` или `false`.

Пример

```
> restart;
> x:=5:
> if x>0 then sin(x)/x else -sin(x)/x fi;
```

$$\frac{1}{5}\sin(5)$$

```
> x:=1:
> `if` (x>0,sin(x)/x,-sin(x)/x);
```

$$\sin(1)$$

В этом случае нужны ` `.

Циклы for, while и do

Maple имеет обобщенную конструкцию цикла, которая задаётся следующим образом:

```
{for <name>} {from <expr1>} {to <expr3>} {by <expr2>}
{while <expr4>}
do <statement> od;
```

`name` – имя управляющей переменной цикла;

`expr1 expr2 expr3` – выражения, задающие начальное, конечное значения и шаг изменения переменной name;

`expr4` – выражение, задающее условие пока цикл (набор объектов между словами `do` и `od`) будет выполняться.

Если блок `by <expr2>` отсутствует, то параметра цикла увеличивается на единицу.

Пример

```
> for i from 1 to 10 by 2 do print(i) od;
```

1
3
5
7
9

Выполнение цикла в обратном порядке (если `expr1>expr2`) не предусмотрено.

Цикл можно прервать с помощью дополнительного блока while <expr4>.

Пример

```
> for i from 1 to 10 by 2 while i<=6 do print(i) od;
                                1
                                3
                                5
```

Таким образом, конструкция цикла в Maple-языке программирования вобрала в себя основные конструкции циклов for и while. Все конструкции могут вкладываться друг в друга.

13.2. Дифференцирование

Вычисление производных функции $f^n(x)=\frac{d\cdot f^n(x)}{d\cdot x^n}$ – одна из самых распространенных задач математического анализа. Для ее реализации СКМ Maple имеет следующие основные функции:

```
diff (a, x1, x2,…, xn),     diff(a, [x1, x2, … , xn]),
Diff (a, x1, x2,…, xn),     Diff(a, [x1, x2, …, xn]).
```

Здесь a – дифференцируемое алгебраическое выражение, в частности функция f(x1, x2,…,xn) ряда переменных, по которым производится дифференцирование.

Функция Diff является инертной формой вычисляемой функции diff и может использоваться для естественного вычисления производной в документах.

В простейшей форме diff(f(x),x) вычисляет первую производную функции f(x) по переменной x. При n, большем 1, вычисления производных выполняются рекурсивно, например, diff(diff(f(x), x), y). Или же для вычисления производных высокого порядка можно использовать оператор $. Например, выражение diff(f(x), x$4), вычисляющее производную четвертого порядка по x, эквивалентно по записи diff(f(x), x, x, x, x).

Пример13.8

```
> Diff(a*x^n,x)=diff(a*x^n,x);
```

$$\frac{\partial}{\partial x} a x^n = \frac{a x^n n}{x}$$

```
> Diff(sin(x),x)=diff(sin(x),x);
```

$$\frac{\partial}{\partial x} \sin(x) = \cos(x)$$

```
> f(x,y):=cos(x)*y^3;
```

$$f(x,y) := \cos(x) y^3$$

```
> Diff(f(x,y),x)=diff(f(x,y),x);
```

$$\frac{\partial}{\partial x}\cos(x)\,y^3 = -\sin(x)\,y^3$$

```
> Diff(f(x,y),x$2,y$2)=diff(f(x,y),x$2,y$2);
```

$$\frac{\partial^4}{\partial y^2\,\partial x^2}\cos(x)\,y^3 = -6\cos(x)\,y$$

13.3. Интегрирование

Вычисление *неопределенного* интеграла обычно заключается в нахождении первообразной функции. Для вычисления неопределенных и определенных интегралов Maple представляет следующие функции:

```
int(f, x),          int(f, x=a..b),
int(f, x = a..b, continuous).
Int(f, x),          Int(f, x=a..b),
Int(f, x=a..b, continuous).
```

Здесь `f` – подынтегральная функция,

`x` – переменная, по которой выполняются вычисления,

`a` и `b` –верхний и нижний пределы интегрирования,

`continuous` – необязательное дополнительное условие.

Для вычисления определенного интеграла необходимо использовать функцию `evalf(int(f, x=a..b))`.

Если верхним пределом интегрирования является бесконечность, то она обозначается словом `infinity`.

Пример 13.9

```
> Int(sin(x)/x,x=0..1.)=int(sin(x)/x, x=0..1.);
```

$$\int_0^{1.} \frac{\sin(x)}{x}\,dx = .9460830704$$

```
> Int(x*exp(-x),x=0..infinity) = int(x*exp(-x),
x=0..infinity);
```

$$\int_0^{\infty} x\,e^{(-x)}\,dx = 1$$

13.4. Типовые средства графики

В само ядро Maple встроено ограниченное число функций графики. Это, прежде всего, функция для построения двумерных графиков `plot` и функция для построения трехмерных графиков `plot3d`. Они позволяют строить графики

наиболее распространенных типов в различных системах координат, как на плоскости, так и в трехмерном пространстве. Для построения графиков более сложных типов необходимо подключать пакеты расширений Maple.

Двумерная графика

Для построения двумерных графиков используется команда `plot`.

Формат

```
plot(function, variable_x {,variable_y}{,option});
```

где `function` – функция, график которой строится;

`variable_x` – переменная, указывающая область изменения по горизонтали;

`variable_y` – переменная, указывающая область изменения по вертикали;

`option` – набор опций, задающий стиль построения графика функции.

При построении графиков функцию можно определять через переменную.

Для двумерной графики можно включать следующие опции:

- *numpoints* – изменение количества точек графика (по умолчанию=49);
- *color* – задание цвета кривой графика;
- *title* – добавление заголовка графика (например, `title="string"`);
- *coords* – выбор системы координат, этот параметр задает 15 типов координатных систем. По умолчанию задана прямоугольная система координат;
- *axes* – задание типа осей координат (*frame* - рамка, *boxed* - прямоугольник, *normal* - ортогональные, *none* – без осей);
- *thickness* – толщина линии графика;
- *xtickmarks, ytickmarks* – управление числом меток на оси, т.е. задает минимальное число отметок по оси x и y соответственно;
- *style* – стиль построения графика (*line* – выводится интерполяционная кривая, *point* – выводятся точки);
- *scalling* – масштаб графика (*constrained* – сжатый, *unconstrained* - несжатый);
- *size* – размер шрифта в пунктах;
- *symbol* – тип точки графика в виде символа (*box* - прямоугольник, *cross* - крест, *circle* - окружность, *point* – точка, *diamond* - ромб);
- *titlefont* – шрифт для заголовка;
- *labelfont* – шрифт для меток (labels) на осях координат;
- *view=[A,B]* – определение максимальной и минимальной координат, в пределах которых график будет отображаться на экране, где A=[xmin..xmax], B=[ymin..ymax].

Пример13.1

Построение графика неявно заданной функции `sin(x)/x` на интервале `-15..15` (см. рис. 13.1).

```
> plot(, x=-15..15,color=red, title="график");
```

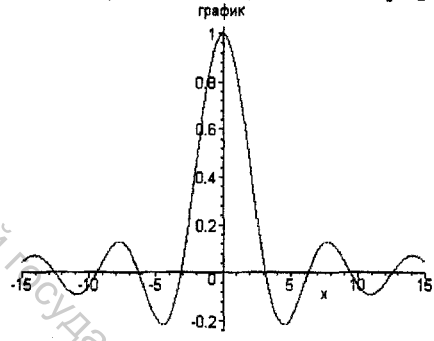


Рис.13.1. График функции `sin(x)/x`

Пример 13.2

Построение графика функции $\sin^2(x)$ определенной с помощью оператора присваивания, на интервале `x=-5..5,y=0..0.5`, черного цвета в виде совокупности точек (см. рис. 13.1).

```
> fun:=sin(x)^2;
```

$$fun := \sin(x)^2$$

```
> plot(fun, x=-5..5,y=0..0.5,color=black,style=point,
title="График");
```

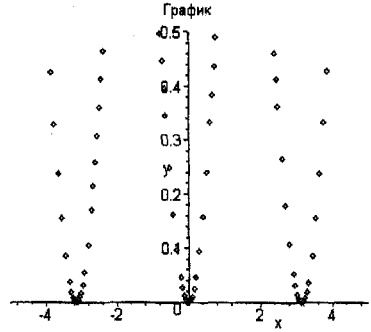


Рис.13.2. График функции `fun=sin^2(x)`

Пример 13.3

Построение графика функции `ln(1+cos(x))`, при x, стремящемся к бесконечности (см. рис. 13.3).

```
>plot(ln(1+cos(x)),x=0..infinity,y=-10..1, color = blue);
```

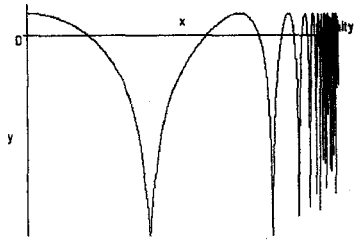


Рис.13.3. График функции `ln(1+cos(x))`

Пример 13.4

Построение графиков трех функций `sin(x)`, `sin(x)/x`, `sin(x^3/100)` линиями трех цветов и трех типов (см. рис.13.4).

```
> plot([sin(x),sin(x)/x, sin(x^3/100)],
x=-10..10,color=[black,blue,red],style=[line,line,point]);
```

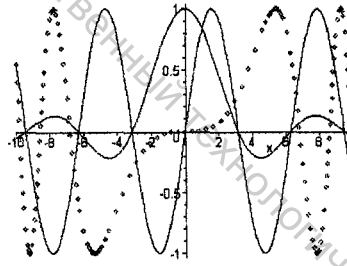


Рис.13.4. График трех функций

Пример 13.5

Построение графика функции, содержащей условное выражение с функцией if (см. рис.13.5).

```
>plot(`if`(sin(x)/x>0,sin(x)/x,-sin(x)/x),x=-10..10,
color=black);
```

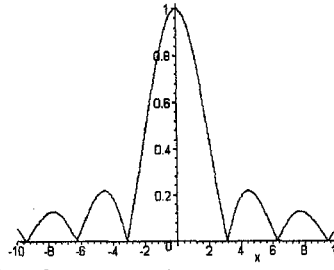


Рис.13.5. График функции, содержащей условное выражение

Пример 13.6

Построение графика функции `1-sin(t)` в полярной системе координат. Здесь существенным моментом является задание полярной системы координат опцией coords=polar.

```
> plot([1-sin(t),t,t=0..6.28],color=blue, style=point,
coords=polar);
```

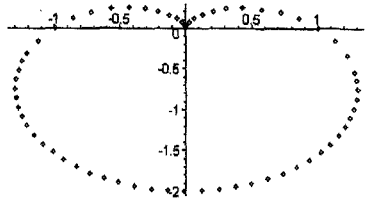


Рис.13.6. График функции в полярной системе координат

Трехмерная графика

Трехмерными называют графики, отображающие функции двух переменных z(x,y). На деле трехмерные графики представляют собой объемные проекты в аксонометрии.

Для построения таких графиков Maple имеет встроенную в ядро функцию `plot3d`. Она может использоваться в следующих форматах:

```
plot3d(expr1, x = a..b, y = c..d, p),
plot3d(f, a..b, c..d, p),
plot3d([exprf, exprg, exprh], s = a..b, t = c..d, p),
plot3d([f, g, h], a..b, c..d, p).
```

Здесь p – параметры, с помощью которых можно в широких пределах управлять видом трехмерных графиков.

Пример 13.7

Построить поверхность h^2 в цилиндрической системе координат (см. рис. 13.7).

```
> plot3d(h^2,a=-Pi..Pi,h=-5..5, coords=cylindrical,
style=patch, color=sin(h));
```

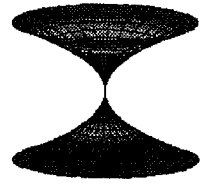


Рис.13.7. Пример трехмерного графика

В последние версии Maple введены новые функции для ускоренного построения графиков. Например, функция `smartplot(f)` предназначена для создания двухмерных графиков, функция `smartplot3d(f)` – для создания трехмерных. Причем функция `smartplot3d(f)` обеспечивает построение не только отдельных поверхностей, но и ряда пересекающихся поверхностей, при этом линии пересечения поверхностей строятся вполне корректно.

13.5. Решение уравнений

Для решения уравнений, неравенств и их систем в СКМ Maple используется функция `solve`, которая возвращает последовательность решений.

Формат

`solve(eqn, var);`

где `eqn` – уравнение, неравенство или процедура;

`var` – имя опции.

Уравнение и его решение можно представлять в виде отдельных объектов, отождествленных с определенной переменной.

Пример 13.8

```
> ur:=x^2+2*x-3;# задание уравнения через переменную ur
> ot:=solve(ur,x);# решение уравнения и присвоение корней переменной ot
> x1:=ot[1];# присвоение первого корня переменной x1
> x2:=ot[2];# присвоение второго корня переменной x2
> subs(x=x1,ur);# подстановка первого корня в уравнение
> subs(x=x2,ur);# подстановка первого корня в уравнение
```

Если решений нет или функция не может найти решение, то возвращается пустая последовательность NULL. В этом случае целесообразно использовать функцию `fsolve`, которая возвращает корень уравнения в форме вещественного числа.

Формат

`fsolve(eqn, var);`

`eqn` – уравнение, неравенство или процедура;

`var` – имя опции.

Пример 13.9

```
> solve(exp(x)+ln(2*x)-4.2*x);
```

$$0.2270523742 - 0.3228803688\,I$$

Как видно из результата решения данного уравнения, корень представлен с использованием мнимой единицы, что не дает представления о его числовом значении, поэтому для его решения следует воспользоваться командой `fsolve`.

```
> fsolve(exp(x)+ln(2*x)-4.2*x);
```

1.886222494

Классическим способом решения нелинейных уравнений является использование одного из численных методов (см. главу 11). Например, сведение уравнения к виду $x = \varphi(x)$ и применение метода простых итераций $x_k = \varphi(x_{k-1})$ при заданном значении x_0.

Пример 13.10 [6]

Методом простой итерации найти корень уравнения `3*ln(x+1)-x = 0`.

Преобразуем это уравнение к виду $x = \varphi(x)$: `x = 3*ln(x+1)`.

Графически определим интервал изоляции корня (см. рис. 13.8):

```
> plot(3*ln(x+1)-x,x=1..7);
```

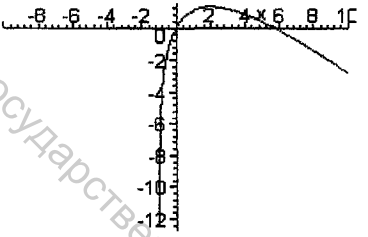


Рис. 13.8. Выделение интервала изоляции корня уравнения

Первый корень, равный нулю, очевиден. Для уточнения второго корня применим метод простой итерации. Для упрощения вычислений положим 16 приближений корня.

```
> x||0:=0.5;
```

$$x0 := .5$$

```
> x0:=0.5;
```

$$x0 := .5$$

```
> for k from 1 to 16 do x||k:=f(x||(k-1)); od;
```

$$x1 := 1.216395324$$

$$x2 := 2.387646445$$

$$x3 := 3.660406248$$

$$x4 := 4.617307866$$

$$x5 := 5.177557566$$

$$x6 := 5.462768931$$

$$x7 := 5.598173559$$

$$x8 := 5.660378631$$

$$x9 := 5.688529002$$

$$x10 := 5.701181910$$

$$x11 := 5.706851745$$

$$x12 := 5.709388956$$

$$x13 := 5.710523646$$

$$x14 := 5.711030964$$

$$x15 := 5.711257755$$

$$x16 := 5.711359134$$

Нетрудно заметить, что значения корня x_k в ходе итераций явно сходятся к некоторому значению. Проведем проверку решения, используя функцию `fsolve`:

```
> f(x)= x;
```

$$3\ln(x+1) = x$$

```
> fsolve((3*ln(x+1)-x),x);
```

$$0., 5.711441084$$

13.6. Решение систем линейных алгебраических уравнений

Системы линейных алгебраических уравнений можно решать также, используя команду `solve`. Такое решение в силу простоты записи может быть предпочтительным. Для решения система уравнений и перечень неизвестных задаются в виде множеств, то есть с использованием фигурных скобок.

Пример 13.11

```
> sys:={3*x1-4*x2-x3=10,6*x1-8*x2-3*x3=19,-x1+x2+x3=-3};
  # задание системы уравнений
> ot:=solve(sys,{x1,x2,x3});# присвоение переменной ot
результатов решения системы
> subs(ot={x1,x2,x3},sys);# подстановка результата в
уравнение
```

$$sys := \{3\,x1 - 4\,x2 - x3 = 10,\ 6\,x1 - 8\,x2 - 3\,x3 = 19,\ -x1 + x2 + x3 = -3\}$$

$$ot := \{x3 = 1,\ x2 = 1,\ x1 = 5\}$$

$$\{3\,x1 - 4\,x2 - x3 = 10,\ 6\,x1 - 8\,x2 - 3\,x3 = 19,\ -x1 + x2 + x3 = -3\}$$

Систему уравнений можно задать с помощью функционального оператора следующим образом:

```
> f:=(a1,a2,a3,b)->a1*x1+a2*x2+a3*x3=b;
> sys:={f(3,-4,-1,10),f(6,-8,-3,19),f(-1,1,1,-3)};
```

$$f := (a1, a2, a3, b) \to a1\,x1 + a2\,x2 + a3\,x3 = b$$

$$sys := \{3\,x1 - 4\,x2 - x3 = 10,\ 6\,x1 - 8\,x2 - 3\,x3 = 19,\ -x1 + x2 + x3 = -3\}$$

Решение систем из трех линейных уравнений имеет наглядную геометрическую интерпретацию – в виде точки, в которой пересекаются три плоскости, каждая из которых описывается функцией двух переменных. Это позволяет сделать функция импликативной графики `implicitplot3d` (см. рис.13.9).

Пример13.12

```
> with(plots):
> sys:={3*x-4*y-z=10,6*x-8*y-3*z=19,-x+y+z=-3}:
> implicitplot3d(sys,x=-6..6,y=-6..6,z=-6..6,color=pink);
```

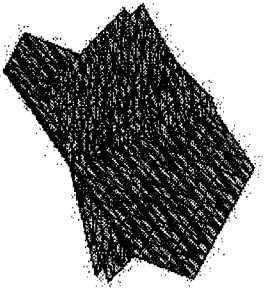


Рис.13.9. Графическое решение системы уравнений

13.7. Задания для самостоятельной работы

- Найти значение первой и второй производных функции f(x):

Вариант	Условие	Вариант	Условие
1	а) $f(x)=2x^4+5x^3+x^7+45$; б) $f(x)=\sin 4x^2$.	6	а) $f(x)=300x^3+500x^{11}+xy^4+7$; б) $f(x)=x\cdot\cos 2x/\sin x$.
2	а) $f(x)=42x^{0,5}+3\sqrt[3]{x}+x^2y+3$; б) $f(x)=\sin x\cdot e^{45x+4}$.	7	а) $f(x)=8x^7+6x^5+4x^3y^2+9$; б) $f(x)=ctg(45x^2+34x+5)+tg(25x)$.
3	а) $f(x)=8x^5+5x^4+3y^6+15$; б) $f(x)=\cos 87x^3$.	8	а) $f(x)=12x^4+52x^2+\frac{y^2}{x}+45x+2$; б) $f(x)=\sin 2x\cdot\cos 2x$.
4	а) $f(x)=23x^{30}+5x^{11}+x\cdot y+45x$; б) $f(x)=\sin^2 x+tg12x\cdot x$.	9	а) $f(x)=44x^4+25x^2+xy^3+4x$; б) $f(x)=\sin(4x+24)\cdot\ln(1-2x)\cdot e^{4x}$.
5	а) $f(x)=\frac{x^{27}}{27}+5x^{0,2}+5x^2y+45x$; б) $f(x)=x\sin x+x^2\cos 43x$.	10	а) $f(x)=5x^{10}+10x^5+25xy+125$; б) $f(x)=tg(2x)+tg(12x)\cdot\cos(8x)$.

- Найти значение определенного интеграла (см. главу 12, п.12.3), дать графическую интерпретацию решения.

- Решить нелинейное уравнение (см. главу 11, п.11.4). Выделить графически интервал изоляции корня уравнения и вычислить корень с точностью $\varepsilon = 10^{-4}$

- Решить систему уравнений и дать графическую интерпретацию:

Вариант	Условие	Вариант	Условие
1	$\begin{cases} 3x+2y+z=5 \\ 2x+3y+z=1 \\ 2x+y+3z=11 \end{cases}$	6	$\begin{cases} 3x-y=5 \\ -2x+y+z=0 \\ 2x-y+4z=15 \end{cases}$
2	$\begin{cases} x+2y+4z=31 \\ 5x+y+2z=29 \\ 3x-y+z=10 \end{cases}$	7	$\begin{cases} 3x-y+z=4 \\ 2x-5y-3z=-17 \\ x+y-z=0 \end{cases}$
3	$\begin{cases} 4x+3y+2z=9 \\ 2x+5y-3z=4 \\ 5x+6y-2z=18 \end{cases}$	8	$\begin{cases} x+y+z=2 \\ 2x-y-6z=-1 \\ 3x-2y=8 \end{cases}$
4	$\begin{cases} 2x-y-z=4 \\ 3x+4y-2z=11 \\ 3x-2y+4z=11 \end{cases}$	9	$\begin{cases} 4x-5y+5z=18 \\ 5x+y-z=8 \\ 6x-4y+3z=21 \end{cases}$
5	$\begin{cases} x+y+2z=-1 \\ 2x-y+2z=-4 \\ 4x+y+4z=-2 \end{cases}$	10	$\begin{cases} 2x-4y+7z=45 \\ 4x-2y+3z=23 \\ 7x-3y+8z=53 \end{cases}$

- Построить поверхность:

Вариант	Условие	Вариант	Условие
1	$f=2\sin(x^2y)$ при x=-2..2, y=-3..3.	6	$f=\sin(x^2+(y-1)^2)$ при x=-2..2, y=-1..3.
2	$f=\frac{1}{\cos(x+y)}$ при x=-4..4, y=-4..4.	7	$f=(e^{xy})^{xy}$ при x=-1..1, y=-1..1.
3	$f=\cos 2x\cdot\sin 5y$ при x=-Pi..Pi, y=-Pi..Pi.	8	$f=\sin((x+2)t)$ при x=-10..10, t=1..20.
4	$f=\sin(xy)$ при x=-Pi..Pi, y=-Pi..Pi.	9	$f=\sin x\cdot\cos x\cdot tg(xy)$ при x=-4..4, y=-4..4.
5	$f=\frac{\sin(x+0{,}6\pi)}{x+11}$ при x = -10..10, y = =-Pi..Pi.	10	$f=\frac{\ln(x+y)}{\cos(x+y)}$ при x=-4..4, y=-4..4.

Вариант 11.

Количество отходов в прядении (у) зависит от обрывности нити (х). Определить вид связи между этими переменными, если $P_{y/x}$=6,29, $P_{x/y}$=0,113.

Вид связи определяется по значению коэффициента корреляции r между переменными, который есть среднее геометрическое коэффициентов регрессии, имеющее их знак, т.е.

$$r = \pm\sqrt{\frac{P_{y/x}}{P_{x/y}}},$$

где $P_{y/x}$ и $P_{x/y}$ – коэффициенты регрессии соответственно у относительно х и х относительно у.

Связь считается слабой, если $|r| < 0{,}3$, если $0{,}3 \le |r| < 0{,}7$ - связь средняя, при $0{,}7 \le |r| \le 1$ связь считается сильной.

Текст программы на языке Pascal:

```
Program Correl;
 Uses CRT;
 Var P_xy,P_yx,r:real;
 Begin
 ClrScr;
 Writeln('Введите значения коэффициентов P_xy и P_yx');
 Readln(P_xy, P_yx);
 r:=sqrt(P_yx/P_xy);
 If r<0.3 Then Writeln('Связь слабая')
          Else If r<0.7 Then Writeln('Связь средняя')
                        Else Writeln('Связь сильная');
 Writeln('r=',r:0:2);
 Delay(5000);
End.
```

Вариант 11.

Индекс производства непродовольственных товаров соответственно году выпуска можно описать следующей зависимостью:

$$y=\begin{cases}2{,}1963x^3-29{,}447x^2+119{,}27x-22{,}167, & x\le 2000\\ -3{,}3x+111{,}3, & 2000<x\le 2002,\\ -5{,}2x^2+20{,}3x+94{,}1, & x>2002\end{cases}$$

где x – год выпуска продукции.

Определить индексы производства непродовольственных товаров за период с 1990 по 2007 год.

Задача сводится к табулированию функции Y на интервале [1...18], где 1 соответствует 1990 г., 2 – 1991 г. , . . . , 18 – 2007 г.

Текст программы на языке Pascal:

```
Program Tab;
 Var x,a,b,god:integer;
               Y:real;
 Begin
 Writeln('Введите границы временного интервала a,b');
 Readln(a,b);
 Writeln(' Год  Индекс');
 x:=1;
 For god:=a To b Do
  Begin
   If x<=11 Then Y:=2.196*exp(3*ln(x))-
29.447*sqr(x)+119.27*x-22.167

             Else If x<=13 Then Y:=-3.3*x+111.3
             Else Y:=-5.2*sqr(x)+20.3*x+94.1;
   Writeln(god:4, ' ':3, Y:0:2);
   x:=x+1;
  End;
 End.
```

Использование подпрограммы-функции

Вариант 7. Определить, используя подпрограмму, годовую экономию текущих затрат по каждому виду работ (расчетных, конструкторских, прочих) и экспериментальному цеху в общем для каждого варианта механизации.

Годовая экономия текущих затрат по i-тому виду работ

$$Э'_{год_i} = L_{год_i} \cdot n_i \cdot \frac{k_{c_i}}{100},$$

где $L_{год_i}$ - годовая заработная плата одного работающего по i-тому виду работ, тыс. руб./год, n_i – количество сотрудников, занятых i-тым видом работ, k_{c_i} - ожидаемое снижение трудоемкости по i-тому виду работ, %.

Должность	Количество	Средняя годовая з/п, тыс. руб.	Вариант 1, $Кс_1$	Вариант 2, $Кс_2$	Вариант 3 $Кс_3$
Инженер-расчетчик	6	1650	30%	10%	50%
Конструктор	75	1750	25%	20%	5%
Служащий	18	1200	40%	25%	12%

Общая годовая экономия текущих затрат по экспериментальному цеху

$$Э'_{год} = \sum_{i=1}^{m} Э'_{год_i} \cdot \left(k_{доп} + \frac{k_н}{100}\right),$$

где m – количество видов работ, $k_{доп.}$ - коэффициент, учитывающий дополнительную заработную плату и начисления, $К_{доп} – 1,3$, $k_н$ – накладные расходы, $К_н – 80$ %.

Текст программы на языке Pascal:

```
program fun1;
uses crt;
const k_dop=1.3; kn=80;
var z_r,n_r,z_k,n_k,z_s,n_s:integer; {Заработная плата и
количество работников}
{Снижение трудоемкости по каждому варианту и виду работ}
    str1,str2,str3:integer;
    stk1,stk2,stk3:integer;
    sts1,sts2,sts3:integer;
{Годовая экономия по каждому варианту и виду работ}
    ekr1,ekr2,ekr3:real;
    ekk1,ekk2,ekk3:real;
    eks1,eks2,eks3:real;
{Общая экономия по вариантам механизации}
```

```
    ob_ek1,ob_ek2,ob_ek3:real;
{Функция для вычисления экономии по каждому виду работ}
function ekon(zp,kol,kc:longint):real;
begin
ekon:=zp*kol*kc/100
end;
{Функция для вычисления общей экономии}
function ob_ekon(ek1,ek2,ek3,k_dop:real;kn:integer):real;
begin
ob_ekon:=(ek1+ek2+ek3)*(k_dop+kn/100)
end;
begin {Начало раздела операторов основной программы}
clrscr;
{Ввод исходных данных}
writeln('Введите з/плату и количество инженеров-
расчетчиков');
read(z_r,n_r);
writeln('Введите з/плату и количество конструкторов');
read(z_k,n_k);
writeln('Введите з/плату и количество служащих');
read(z_s,n_s);
writeln('Введите снижение трудоемкости по 1 варианту для
всех работ');
read(str1,stk1,sts1);
writeln('Введите снижение трудоемкости по 2 варианту для
всех работ');
read(str2,stk2,sts2);
writeln('Введите снижение трудоемкости по 3 варианту для
всех работ');
read(str3,stk3,sts3);
writeln;
{Расчет годовой экономии для 1 варианта механизации}
ekr1:=ekon(z_r,n_r,str1);
ekk1:=ekon(z_k,n_k,stk1);
eks1:=ekon(z_s,n_s,sts1);
{Расчет годовой экономии для 2 варианта механизации}
ekr2:=ekon(z_r,n_r,str2);
ekk2:=ekon(z_k,n_k,stk2);
eks2:=ekon(z_s,n_s,sts2);
{Расчет годовой экономии для 3 варианта механизации}
ekr3:=ekon(z_r,n_r,str3);
ekk3:=ekon(z_k,n_k,stk3);
eks3:=ekon(z_s,n_s,sts3);
{Расчет общей экономии по всем вариантам механизации}
ob_ek1:=ob_ekon(ekr1,ekk1,eks1,k_dop,kn);
ob_ek2:=ob_ekon(ekr2,ekk2,eks2,k_dop,kn);
ob_ek3:=ob_ekon(ekr3,ekk3,eks3,k_dop,kn);
writeln('Экономия по расчетным работам по вариан-
там=',ekr1:6:1,' ',ekr2:6:1,' ',ekr3:6:1);
```

```
writeln('Экономия по конструкторским работам по вариан-
там=',ekk1:6:1,' ',ekk2:6:1,' ',ekk3:6:1);
writeln('Экономия по прочим работам по вариан-
там=',eks1:6:1,' ',eks2:6:1,' ',eks3:6:1);
writeln;
writeln('Общая экономия по 1 варианту=',ob_ek1:6:1);
writeln('Общая экономия по 2 варианту=',ob_ek2:6:1);
writeln('Общая экономия по 3 варианту=',ob_ek3:6:1);
readln
end.
```

Сеанс работы с программой[3]:

Введите з/плату и количество инженеров-расчетчиков
1650 6
Введите з/плату и количество конструкторов
1750 75
Введите з/плату и количество служащих
1200 18
Введите снижение трудоемкости по 1 варианту для всех работ
30 25 40
Введите снижение трудоемкости по 2 варианту для всех работ
10 20 25
Введите снижение трудоемкости по 3 варианту для всех работ
50 5 12

Экономия по расчетным работам по вариантам=2970.0 990.0 4950.0
Экономия по конструкторским работам по вариантам=32812.5 26250.0 6562.5
Экономия по прочим работам по вариантам=8640.0 5400.0 2592.0

Общая экономия по 1 варианту=93287.2
Общая экономия по 2 варианту=68544.0
Общая экономия по 3 варианту=29619.4

При решении задачи значения коэффициентов снижения трудоемкости возможно представить в виде матрицы, а количество работников и их заработную плату – в виде векторов.

Использование подпрограммы-процедуры

Вариант 10. Определить рациональную мощность потока по изготовлению женских блузок, если такт потока, τ=75 с., затрата времени на единицу, T=5574 с., продолжительность смены, R=28800 с.

[3] Здесь и далее в приложениях курсивом обозначен текст, выводимый компьютером.

Для выбора рациональной мощности рассчитываются следующие технико-экономические показатели при отклонениях от такта потока 80, 90, 100, 105 и 110 %. Расчет показателей оформить с использованием подпрограмм.

1. Расчетный такт потока $\tau_p = \frac{\%откл}{100} \cdot \tau$.

2. Мощность потока в смену, ед., $M = \frac{R}{t_{p_i}}$. Полученное значение округлить до целого.

3. Расчетное количество рабочих $N_p = \frac{T}{t_{p_i}}$.

4. Фактическое количество рабочих Nф принимается округлением расчетного количества до целого.

5. Коэффициент использования рабочего времени $K_{исп} = \frac{N_p}{N_\phi}$.

6. Производительность труда 1 рабочего $П_T = \frac{M}{N_\phi}$.

Мощность рациональна, если $K_{исп}$=1, производительность труда больше средней, а рекомендуемое количество рабочих составляет 50-90 человек.

Текст программы на языке Pascal:

```
program proc1;
uses crt;
const T=5574; R=28800; takt=75;
var i,pr:integer;
    takt_r,N_r,K_isp,P_t:real;
    M,N_f:integer;
{Процедура вычисления технико-экономических показателей}
procedure pokaz(otkl:integer; var r_t,Nr,Kisp,Pt:real; var
M,Nf:integer);
begin
r_t:=otkl/100*takt;
M:=round(R/r_t);
Nr:=T/r_t;
Nf:=round(Nr);
Kisp:=Nr/Nf;
Pt:=M/Nf
end;
begin {Начало раздела операторов основной программы}
clrscr;
{Организация цикла для расчета ТЭП}
for i:=1 to 5 do
begin
writeln('введите процент отклонения');
readln(pr);
```

```
pokaz(pr,takt_r,N_r,K_isp,P_t,M,N_f);
writeln('показатели для отклонения ',pr:2,'%');
write(takt_r:4:1,' ',N_r:3:1,' ',K_isp:3:1,' ',P_t:3:1,'
',M,' ',N_f);
if frac(K_isp)<0.1 then K_isp:=int(K_isp); {Так как в Пас-
кале нет функции округления с точностью до знака, проверя-
ется значение дробной части. Если оно меньше 0,1, то K_isp
округляется до целого}
if (K_isp=1) and (N_f>=50) and (N_f<=90) then writeln('
Мощность рациональна')
else writeln(' Мощность нерациональна') {Проверка на рацио-
нальность}
end;
readln
end.
```

При решении задачи нет необходимости вводить переменные для каждого отклонения от такта потока, так как предусмотрена возможность вывода значений внутри цикла. Возможно решение этой задачи с использованием массивов для хранения в них значений технико-экономических показателей потока.

Сеанс работы с программой:

Введите процент отклонения
80
Показатели для отклонения 80%
60.0 92.9 1.0 5.2 480 93 Мощность нерациональна
Введите процент отклонения
90
Показатели для отклонения 90%
67.5 82.6 1.0 5.1 427 83 Мощность нерациональна
Введите процент отклонения
100
Показатели для отклонения 100%
75.0 74.3 1.0 5.2 384 74 Мощность рациональна
Введите процент отклонения
105
Показатели для отклонения 105%
78.8 70.8 1.0 5.2 366 71 Мощность нерациональна
Введите процент отклонения
110
Показатели для отклонения 110%
82.5 67.6 1.0 5.1 349 68 Мощность нерациональна

ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ ОБРАБОТКИ МАССИВОВ

Обработка одномерного массива

Вариант 1. Сведения о количестве проданных товаров каждым продавцом представлены в виде целочисленного вектора из 20 элементов. Найти количество продавцов, объем продаж у которых больше нуля. Перед вводом объема продаж каждого продавца должна выводиться подсказка с номером продавца (1, 2, 3 и т.д.).

Вывести на экран исходный массив и количество продавцов.

Данные для отладки:

Pr = {20; 0; 30; 23; 34; 0; 10; 9; 27; 45; 0; 24; 35; 17; 29; 24; 16; 34; 0; 8}.

Текст программы на языке Pascal:

```
Program Prodazy;
 Uses Crt;
 Var Pr:array[1..20] of integer;
     i, kolich:integer;
 Begin
 Writeln('Введите массив продаж');
 For i:=1 To 20 Do
  Begin
   Writeln('Введите значение для', i:2,'-го продавца');
   Readln(Pr[i]);
  End;
 Kolich:=0;
 For i:=1 To 20 Do
  If Pr[i]>0 Then Kolich:=Kolich+1;
 Writeln('Список продаж');
 For i:=1 To 20 Do
 Write(Pr[i]:3);
 Writeln;
 Writeln('Продажи совершили ',kolich:2,' человек');
 Delay(5000);
 End.
```

Обработка двумерного массива

Вариант 15. В магазине установлены скидки за покупку товаров: 5% - за покупку товаров на сумму, превышающую 20000 рублей и 3% - за покупку товаров в ночное время. В таблице представлены сведения о сумме покупок, сумме покупок со скидками 5 % и 3 %. Сформировать вектора пятипроцентной скидки и скидки 3% за покупку товаров в ночное время. В каждом векторе найти минимальную и максимальную сумму скидки.

Сумма покупки	Со скидкой 5%	Со скидкой 3% за покупки в ночное время
20 150	19 143	19 143
8 750	8 750	8 750
22 600	21 470	21 470
21 360	20 292	20 292
5 250	5 250	5 093
12 896	12 896	12 509
24 070	22 867	22 181

Исходную матрицу, вектора скидок, минимальные и максимальные значения скидок вывести на экран. При решении задачи возможно использование подпрограмм.

Скидка = Сумма покупки – Сумма покупки со скидкой (3% или 5 %).

Текст программы на языке Pascal:

```
program skidki;
uses crt;
type matr=array[1..7,1..3] of integer;
     mas=array[1..7] of integer;
var tab:matr;
    sk5,sk3:mas;
    i,j:integer;
    min5,min3,max5,max3:integer;
{Процедура вывода вектора}
procedure vyvod(n:integer;var v:mas);
begin
for i:=1 to n do
write(v[i],' ');
writeln
end;
{Процедура формирования вектора}
procedure form(m,k:integer;w:matr; var v:mas);
begin
  for i:=1 to m do
     for j:=1 to k do
```

```
      v[i]:=w[i,1]-w[i,k]
end;
{Процедура поиска минимального в векторе}
procedure min(n:integer;v:mas; var m:integer);
begin
m:=v[1];
for i:=1 to n do
if v[i]<m then m:=v[i]
end;
{Процедура поиска максимального в векторе}
procedure max(n:integer;v:mas; var m:integer);
begin
m:=v[1];
for i:=1 to n do
if v[i]>m then m:=v[i]
end;
begin
clrscr;
writeln('Ввод исходных данных');
for i:=1 to 7 do
  begin
  write('Введите ',i,' строку данных=>');
    for j:=1 to 3 do
    read(tab[i,j])
  end;
form(7,2,tab,sk5); {Формирование вектора 5% скидок}
form(7,3,tab,sk3); {Формирование вектора 3% скидок}
min(7,sk5,min5); {Поиск минимального среди 5% скидок}
min(7,sk3,min3); {Поиск минимального среди 3% скидок}
max(7,sk5,max5); {Поиск максимального среди 5% скидок}
max(7,sk3,max3); {Поиск максимального среди 3% скидок}
{Вывод исходной таблицы}
writeln('Исходная таблица');
for i:=1 to 7 do
  begin
    for j:=1 to 3 do
    write(tab[i,j]:5,' ');
  writeln
  end;
writeln('вектор скидок 5%');
vyvod(7,sk5);
writeln('вектор скидок 3%');
vyvod(7,sk3);
writeln('минимальное среди скидок 5%=',min5:5);
writeln('минимальное среди скидок 3%=',min3:5);
writeln('максимальное среди скидок 5%=',max5:5);
writeln('максимальное среди скидок 3%=',max3:5);
readln
end.
```

Сеанс работы с программой:

Ввод исходных данных
Введите 1 строку данных=>20150 19143 19143
Введите 2 строку данных=>8750 8750 8750
Введите 3 строку данных=>22600 21470 21470
Введите 4 строку данных=>21360 20292 20292
Введите 5 строку данных=>5250 5250 5250
Введите 6 строку данных=>12896 12896 12509
Введите 7 строку данных=>24070 22867 22181
Исходная таблица
20150 19143 19143
8750 8750 8750
22600 21470 21470
21360 20292 20292
5250 5250 5250
12896 12896 12509
24070 22867 22181
вектор скидок 5%
1007 0 1130 1068 0 0 1203
вектор скидок 3%
1007 0 1130 1068 0 387 1889
минимальное среди скидок 5%= 0
минимальное среди скидок 3%= 0
максимальное среди скидок 5%= 1203
максимальное среди скидок 3%= 1889

ЛИТЕРАТУРА

1. Абрамов, В. Введение в язык Паскаль / В. Г Абрамов, Н. П. Трифонов, Г. Н. Трифонова. – Москва : Наука, 1988. – 320 с.
2. Буза, М. Windows-приложения: от операции к реализации / М. К. Буза, Л. В. Певзнер. – Минск : Вышейшая школа, 1998. – 490 с.: ил.
3. Вальвачев, А. Программирование на языке Паскаль для персональных ЭВМ ЕС / А. Н Вальвачев, В. С. Крисевич. – Минск : Вышейшая школа, 1991. – 224 с.
4. Вычислительная техника и программирование / под ред. А. В. Петрова. – Минск : Вышейшая школа, 1991. – 479 с.
5. Грогоно, П. Программирование на языке Паскаль / П. Грогоно – Москва : Мир, 1982. – 268 с.
6. Дьяконов, В. Maple 6 : учебный курс / В. Дьяконов. – Санкт-Петербург : Питер, 2001. – 608 с. : ил.
7. Епанешников, А. Программирование в среде Turbo Pascal 7.0 / А. Епанешников, В. Епанешников. – Москва : Диалог – МИФИ, 2000. – 256 с.
8. Информационные технологии в экономике / А. Н. Романов. – Минск : ЗАО «Веды», 1998. – 240 с.
9. Йенсен, К. ПАСКАЛЬ: руководство для пользователя и описание языка / К. Йенсен, Н. Вирт. – Москва : Финансы и статистика, 1982. – 312 с.
10. Першиков, В. И. Толковый словарь по информатике / В. И. Першиков, В. М. Савинков. – Москва : Финансы и статистика, 1991. – 543 с.
11. Прикладная информатика : учебное пособие / А. Н. Морозевич [и др.] ; под общ. ред. А. Н. Морозевича. – Минск : Вышейшая школа, 2003. – 335 с.: ил.
12. Семашко, Г. Л. Программирование на языке Паскаль / Г. Л Семашко, А. И. Салтыков. – Москва : Наука, 1988. – 118 с.
13. Статистический ежегодник Республики Беларусь, 2004 / Министерство статистики и анализа Республики Беларусь. – Минск, 2004. – 660 с.
14. Турчак, Л. И. Основы численных методов / Л. И. Турчак – Москва : Наука, 1987. – 328 с.

Учебное издание

Вардомацкая Елена Юрьевна

Окишева Татьяна Николаевна

Информатика в прикладных задачах легкой промышленности

Пособие

Редактор В. Е. Казаков
Технический редактор О. П. Колоскина
Корректор Т. К. Покатович
Компьютерная верстка Е. Л. Афанасьева

Подписано к печати 23.03.07. Формат 60x84 1/16. Бумага офсетная №1.
Гарнитура «Таймс». Усл.-печ. листов 12,8. Уч.-издат. листов 11,9. Тираж 313 экз. Зак. № 155

Учреждение образования «Витебский государственный технологический университет» 210035, г. Витебск, Московский пр-т, 72.

Отпечатано на ризографе учреждения образования «Витебский государственный технологический университет»
Лицензия №02330/0133005 от 1 апреля 2004 г.