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

Е.Ю. Вардомацкая, Т.Н. Окишева

ИНФОРМАТИКА

В двух частях
Часть I

Допущено Министерством образования Республики Беларусь в качестве учебного пособия для студентов учреждений, обеспечивающих получение высшего образования по специальностям легкой промышленности

Витебск
2007

УДК 004.9(075.8)
ББК 32.81
В 18

Авторы: Вардомацкая Е.Ю.
Окишева Т.Н.

Рецензенты:
кафедра «Полиграфическое оборудование и системы обработки информации» Белорусского государственного технологического университета (заведующий кафедрой кандидат технических наук доцент Юденков В. С.);
кандидат физико-математических наук доцент кафедры «Основы машиностроительного производства и профессионального обучения» Белорусского национального технического университета Кравченя Э. М.

Вардомацкая, Е.Ю.
В 18 Информатика. В двух частях. Часть I : учебное пособие. / Е. Ю. Вардомацкая, Т. Н. Окишева. – Витебск: УО «ВГТУ», 2007. – 220 стр.

ISBN 985-481-083-6

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 985-481-083-6

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

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 соответствующих вузов введены такие курсы как «Информатика», «Информатика, численные методы и компьютерная графика», «Основы информатики и вычислительной техники». Будущий специалист должен не только иметь представление о состоянии, возможностях и перспективах применения персональных компьютеров, но и уметь 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 одном из языков программирования, эффективно использовать пакеты прикладных программ для решения различных инженерно-технических и экономических задач.

Данное пособие направлено на изучение разделов вышеназванных курсов в соответствии с действующими учебными программами и призвано помочь студентам закрепить полученные знания на конкретном материале, а также получить практические навыки решения задач, связанных с будущей специальностью, средствами современных компьютерных технологий.

Настоящее пособие состоит из двух частей. Часть I раскрывает базовые понятия курса информатики: технические и программные средства персональных компьютеров, основы алгоритмизации, основы программирования на алгоритмическом языке, компьютерное моделирование инженерно-технологических задач. Часть II посвящена анализу и описанию возможностей использования пакетов прикладных программ для решения инженерно-технических, финансовых и экономических задач.

Пособие состоит из шестнадцати глав. Каждая глава включает теоретический материал, содержащий определения и разъяснения основных понятий, и задания для самостоятельной работы, представленные либо в виде вопросов по данной теме, либо в виде прикладных задач различного уровня сложности.

Глава 1 «Введение в предметную область» раскрывает основные понятия информатики и дает общие сведения об изучаемой дисциплине.

В главе 2 «Основы алгоритмизации инженерно-технических задач» представлены сведения о способах описания алгоритмов, а также алгоритмические конструкции, используемые при решении инженерно-технических и экономических задач. Авторы сознательно не представляют в данном разделе примеры построения блок-схем алгоритмов различных вычислительных процессов, так как в дальнейшем эти примеры приводятся при рассмотрении соответствующих конструкций языка программирования Паскаль.

В главах 3 и 4 описаны состав, архитектура персонального компьютера и его программное обеспечение.

В главе 5 «Текстовые процессоры» представлена классификация текстовых процессоров и рассмотрены возможности их использования для организации документооборота на примере текстового процессора MS Word.

Главы 6-9 раскрывают базовые понятия языка программирования Паскаль. В них рассмотрены основные языковые конструкции, используемые при решении инженерно-технических, экономических или финансовых задач.

Глава 10 «Подпрограммы в языке Паскаль» отражает сущность модульного программирования в Паскале. Особое внимание уделяется организации подпрограмм пользователя, объясняется логика формирования процедур и функций пользователя, а также отличия одного вида подпрограмм от другого.

Главы 11-12 посвящены структурированным типам данных языка Паскаль, таким как массивы, строки, записи.

В главе 13 «Сортировка и поиск» рассмотрены методы упорядочивания данных структурированных типов по какому-либо признаку, а также принципы организации поиска в упорядоченном и неупорядоченном наборах данных.

Главы 14-16 посвящены компьютерному моделированию инженерно-технических задач и раскрывают вопросы использования элементов математического анализа, реализуемых средствами языка программирования Паскаль и системы компьютерной математики Maple, для решения различных задач.

При изложении материала внимание уделяется как теории, так и практике применения предлагаемых методик расчетов в различных отраслях легкой промышленности, экономико-организационным постановкам и моделированию задач. Предлагаемые примеры и задания основаны на практических материалах, собранных авторами на предприятиях Республики Беларусь, или представленных в ряде специальных изданий.

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 подготовке курсовых и дипломных работ.

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

Учебное пособие написано на основе лекций, читаемых авторами студентам экономического и художественно-графического факультетов Витебского государственного технологического университета.

Авторы считают своим долгом выразить глубокую благодарность и признательность уважаемым рецензентам за ценные конструктивные замечания, способствующие повышению научно-методического уровня рукописи, а также редакционно-издательскому совету Витебского государственного технологического университета, оказавшему неоценимую помощь и поддержку в процессе работы над пособием.

ГЛАВА 1. ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ

1.1. Понятие информатики. Информационные процессы

Обучение основам информатики и формирование общей информационной культуры является неотъемлемой частью подготовки современного специалиста, способного принимать профессионально грамотные решения в стремительно изменяющемся мире XXI века. Без использования новейших инновационных технологий сегодня немыслимо ни одно производство.

Термин «информатика» возник в 60-х годах во Франции для названия области, занимающейся автоматизированной обработкой информации с помощью компьютера. Этот термин образован путём слияния слов «information» (информация) и «automatigus» (автоматика) и означает «информационная автоматика или автоматизированная обработка информации». Поэтому понятие информатики неразрывно связано с понятием информации.

Информация – это сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, которые уменьшают имеющуюся о них степень неопределённости, неполноты знания [7].

Информатика – это наука о законах и методах хранения, обработки и передачи информации с помощью компьютера. Она базируется на использовании вычислительной техники, дает знания о законах, методах и способах организации информационных процессов в природе и обществе. Информатика находится на стыке нескольких наук – математики, физики, микроэлектроники, статистики, экономики. Она рассматривает информацию как концептуально связанные между собой сведения, данные, понятия, изменяющие наши представления о явлении или объекте окружающего мира.

Информация может классифицироваться по различным критериям.

- 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 деятельности.

Постоянная (условно-постоянная) – неизменная и многократно используемая в течение длительного периода времени информация. Может быть *справочной* (описание свойств объекта в виде устойчивых длительное время признаков); *нормативной* (местные, отраслевые и общегосударственные нормативы), *плановая* (многократно используемые 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 – визуальная информация, представленная в форме графиков, рисунков, схем, чертежей, фотографий, картин и т.п.

- Функции управления.

По функциям управления 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 объекта за определённый период времени.

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

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.

Информационным процессом называется процесс, целью которого является передача, хранение или обработка информации. Процесс переноса информации от одного объекта к другому называется *передачей* (приемом) информации [8].

В процессе передачи информации можно выделить *выдачу* информации (источником) и *получение* информации (приемником).

Если один приемник получает информацию от нескольких источников, то такой процесс называется *сбором информации*. Например, эксперт проводит

анкетирование по какой-либо проблеме. В этом случае эксперт выступает приемником информации, а анкетируемые – источником.

Если источник информации один, а получателей информации несколько, то такой процесс называется *распространением* информации. Примером этого процесса может служить чтение лекций студентам: преподаватель (источник) рассказывает материал (выдает информацию) по определенной теме, а студенты (приемники) его слушают.

Зачастую возникают ситуации, когда источник и приемник обмениваются информацией друг с другом. Такой процесс называется обменом информации. Например, два друга обсуждают интересующее их событие. В этом случае беседа строится в форме диалога, и оба рассказчика выступают и как источники, и как приемники.

1. 2. Средства обработки информации

Для обработки информации используются различные средства. К ним относятся не только технические средства, но и методы обработки информации. Технические средства обработки информации делятся на две группы: *вспомогательные*, используемые человеком для облегчения работы по обработке информации, и *автоматические*, способные обрабатывать информацию без участия человека.

К вспомогательным средствам обработки информации можно отнести микрокалькуляторы, которые, хотя и позволяют выполнять вычисления различной сложности, но все же требуют участия человека.

Типичным представителем второй группы средств обработки информации является электронно-вычислительная машина. Этот термин использовался ранее для обозначения программно управляемой среды, предназначенной для восприятия, хранения, обработки и передачи информации. Сейчас его заменил термин «персональный компьютер» (ПК), под которым понимается ЭВМ, предназначенная для индивидуального применения, ориентированная на решение широкого круга задач, как при автономном использовании, так и в компьютерных сетях.

Минимальной единицей представления информации в компьютере является бит. *Бит* – это один из двух символов, используемых для представления данных в двоичной системе счисления, например, 0 или 1, ДА или НЕТ, True или False и т. д. Хранение, передача и обработка информации в компьютере осуществляется *байтами* – единицей информации, имеющей 2^8=256 значений от 0 до 255. 1 байт = 8 бит.

256 значений байта хватает на кодирование достаточно большого числа знаков, например, всех знаков русского или английского языков. Каждая по-

следующая единица измерения объема информации, например, килобайт, мегабайт и т. д., получается умножением предшествующей единицы в 1024 раза.

1.3. Классификация персональных компьютеров

Класс персональных компьютеров складывается из разнообразных видов ПК, которые могут отличаться друг от друга внешним видом, массогабаритными показателями, объемом оперативной памяти, типом процессора, емкостью жесткого диска и пр. Классификация персональных компьютеров достаточно условна, т.к. комплектация машин и даже конкретные модели подгоняются под требования заказчика.

По массо-габаритным показателям персональные компьютеры можно классифицировать как:

- *настольные* (Desktop) – имеют массу 5-10 кг, используются для оборудования рабочих мест и обладают широким набором функциональных возможностей;
- *носимые* – имеют массу менее 5 кг. К ним относятся: *переносные* (LapTop), предназначенные для использования в поездках и имеющие широкий набор возможностей, в том числе и подключение к компьютерным сетям; *блокнотные* (Notebook), также используемые в поездках, однако с сокращенным набором функциональных возможностей; *электронные секретари* (PDA – Personal Digital Assistant и PalmTop – «лежащий на ладони») – карманные переносные компьютеры, набор функций которых позволяет выполнять записи текстов, некоторые вычисления, составлять расписание.

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. К *учебным* относятся компьютеры среднего класса, предназначенные для автоматизации учебного процесса. *Профессиональные* компьютеры предназначены для решения задач в различных областях. На их основе создаются автоматизированные рабочие места.

1.4. 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 несколько этапов.

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-схемы 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 задачи 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.

Эти этапы выполняются последовательно, один за другим. Ошибка на предыдущем этапе может привести к получению некорректного результата, даже если в тексте программы нет ошибок. Поэтому при подготовке задачи к решению на ПК большое внимание уделяется именно начальным этапам: постановке задачи и ее математической формулировке, выбору метода решения и составлению алгоритма решения задачи.

1.5. Вопросы для самоконтроля

1. Дайте определение и поясните понятие *«информация»*.
2. Дайте определение и поясните понятие *«информатика»*.
3. По каким признакам может классифицироваться информация?
4. В каких единицах измеряется информация?
5. Сколько байт в 1 килобайте?
6. Перечислите виды информационных процессов.
7. Дайте определение и охарактеризуйте процесс сбора информации.
8. Дайте определение и охарактеризуйте процесс распространения информации.
9. Дайте определение и охарактеризуйте процесс обмена информацией.
10. В каком случае данные являются информацией? Приведите пример.
11. Дайте определение и поясните понятие «средства обработки информации».
12. На какие группы подразделяются технические средства обработки информации?
13. Можно ли отнести к техническим средствам обработки информации микрокалькулятор? Если да, то к какой группе?
14. Дайте определение «персональный компьютер». К какой группе средствам обработки информации он относится?
15. По каким признакам и как могут классифицироваться ПК?
16. К какому классу можно отнести компьютер, если его хозяин – профессиональный программист, а компьютер находится дома?
17. Перечислите этапы подготовки к решению задачи на ПК.
18. Что представляет собой этап отладки программы?
19. Какие виды ошибок могут быть выявлены при отладке программы?
20. Допустимо ли изменение порядка выполнения этапов подготовки к решению задачи на ПК? Обоснуйте ответ.

ГЛАВА 2. ОСНОВЫ АЛГОРИТМИЗАЦИИ ИНЖЕНЕРНО-ТЕХНИЧЕСКИХ ЗАДАЧ

2.1. 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 этих устройств последовательность действий должна быть четко сформулирована. Иначе говоря, исполнителю должен быть указан алгоритм решения задачи, представленный на понятном ему языке. Под исполнителем подразумевается как человек, так и компьютер.

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 с 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 выполнения арифметических действий, 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, понятностью, результативностью, массовостью.

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.

2.2. 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.). Эти алгоритмы предназначены для человека, поэтому в качестве команд могут использоваться привычные для человека предложения, фразы. Форма записи должна четко определять 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).

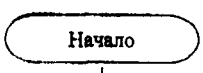
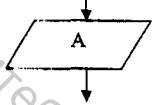
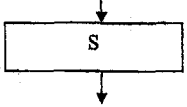
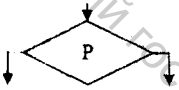
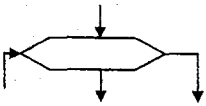
Правила выполнения схем алгоритмов регламентирует ГОСТ 19.002-80 (соответствует международному стандарту ISO 2636-73).

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.

Часто используются блоки, представленные в таблице 1.1.

При значительном количестве пересекающихся линий, большой их длине и многократных изменениях направления допускается разрывать линии потока информации, размещая на обоих концах разрыва *соединитель*.

Таблица 1.1. Блоки, используемые при построении блок-схем

	Начальный блок		Конечный блок
	Блок ввода – вывода. Используется для обозначения операций обмена информацией между устройствами.		Блок вычислений или функциональный блок. В этом блоке осуществляется преобразование информации по заданному действию S.
	Блок проверки условия используется для управления преобразованием информации. В результате анализа выполнения условия P выбирается одно из двух возможных направлений «да» или «нет», и управление передается блоку, записанному на выбранном направлении.		Блок модификации. Используется для организации цикла с параметром. Сначала управление передается к блоку по стрелке вниз, который повторяется заданное число раз, затем управление передается к блоку, следующему за блоком модификации.
	Соединитель		Межстраничный 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.

Представление алгоритма 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*.

2.3. Основные управляющие структуры

При изображении алгоритмов при помощи схем используются базовые управляющие структуры: *следование, развилка, повторение* (рис. 2.1, 2.2, 2.3).

Следование – структура, означающая, что действия S1 и S2 должны быть исполнены одно за другим. В качестве действий S1 и S2 могут выступать любые управляющие структуры или их последовательности.

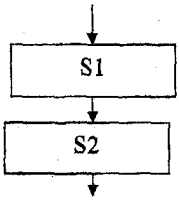


Рис. 2.1. Структура «следование»

Примером использования структуры «следование» на производстве является технологическая последовательность на изготовление изделия. Здесь каждое действие представлено в виде технологически неделимой операции, а сама последовательность есть порядок выполнения этих операций. Например, Вы не сможете проложить отделочную строчку по воротнику, предварительно его не собрав.

Развилка – это действие, 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*

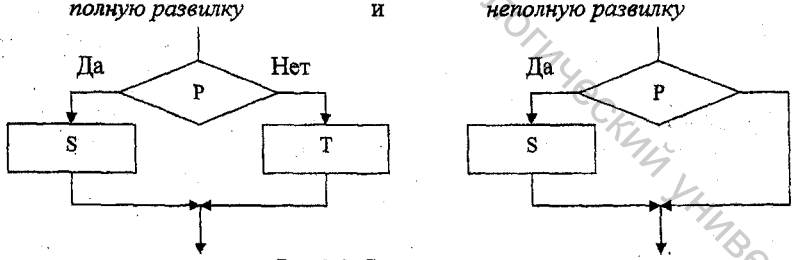


Рис 2.2. Структура «развилка»

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

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

На производстве полной развилкой можно описать процесс определения качества изделия. Если изделие отвечает требуемым показателям, то оно является качественным, соответственно его можно пускать в продажу. Если изделие

не отвечает предъявляемым требованиям, то оно считается бракованным, и его необходимо переделать. Неполную развилку можно представить на таком примере. Если в модели брюк есть накладной карман, то его необходимо настрочить на требуемое место, иначе следует перейти к операции, следующей за настрачиванием кармана.

Повторение – это структура, описывающая циклические вычислительные процессы. Различают *цикл-пока и цикл-до* (см. рис. 2.3).

Цикл-пока словесно можно описать так: 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.

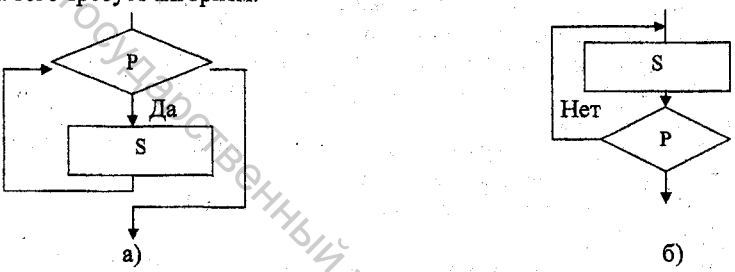


Рис. 2.3. Структура повторение: а) цикл-пока, б) цикл-до

Структурой «повторение» можно описать процесс изготовления изделия на поточной линии. Каждый рабочий выполняет определенную операцию. На потоке изготавливается несколько изделий, следовательно, рабочий должен повторить выполнение заданной операции столько раз, сколько изделий будет изготовлено. Этот процесс продолжается до тех пор, пока не наступит перерыв (цикл-до). Если описать этот процесс циклом-пока, то условие продолжения цикла будет таким: процесс выполнения операции будет продолжаться, пока время работы рабочего не равно времени смены. В данном примере опущено время на обед или личные нужды рабочего, и процесс представлен укрупненно.

2.4. Вопросы для самоконтроля

1. Что означает понятие *«алгоритм»?*
2. Перечислите и поясните свойства алгоритмов.
3. Перечислите и охарактеризуйте способы описания алгоритмов.
4. Приведите примеры алгоритмов в математике.
5. Перечислите блоки, используемые для составления блок-схем.
6. Перечислите основные требования к составлению блок-схем.

7. Объясните, в каких случаях используется блок *«соединитель»*, а в каких случаях *«межстраничный соединитель»*.

8. В каких случаях допускается не указывать направление линии потока информации на блок-схеме?

9. Охарактеризуйте базовую алгоритмическую структуру *«следование»*.

10. Охарактеризуйте базовую алгоритмическую структуру *«развилка»*.

11. Какие блоки используются при изображении структуры *«развилка»*? Охарактеризуйте их.

12. В каком порядке нумеруются блоки на блок-схеме при использовании структуры *«развилка»*?

13. Какое действие выполняется, если в неполной развилке условие P ложно?

14. Какие действия будут выполняться, если в полной развилке условие P истинно, а действие S представляет собой цикл-пока?

15. Охарактеризуйте базовую алгоритмическую *«повторение»*.

16. Какие блоки используются при изображении структуры *«повторение»*? Охарактеризуйте их.

17. Может ли структура «повторение» быть вложена в структуру «развилка»? Приведите примеры вложенных структур.

18. Сколько раз повторится выполнение тела цикла S в цикле-до, если условие истинно?

19. Какое действие выполнится, если условие P в цикле-пока ложно?

20. Какое действие выполнится, если условие P в цикле-до ложно?

ГЛАВА 3. СОСТАВ И АРХИТЕКТУРА ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

3.1. Состав персонального компьютера

В настоящее время в мире известно несколько различных систем построения компьютеров, которые называются *стандартами*. Компьютеры разных стандартов несовместимы между собой. Это означает, что программы, написанные для компьютера одного стандарта, не могут быть использованы на компьютере другого стандарта. Несовместимость базируется на том, что цифровые коды команд для процессоров различаются.

Первый стандарт был создан американской фирмой Apple Computer, которая производит компьютеры марки Macintosh на базе процессора фирмы Motorolla. Стандарт получил название Mac.

Фирмой IBM (International Business Mashines), использующей процессоры фирмы Intel, был разработан другой стандарт. Такие компьютеры называют компьютерами стандарта PC (IBM-совместимыми). Это связано с тем, что первый компьютер фирмы IBM назывался IBM PC (PC – Personal Computer).

В компьютерах типа PC в состав базового комплекта входят системный блок, монитор (дисплей), клавиатура, манипулятор «мышь». Основные составляющие компьютера получили название *аппаратных средств*.

Архитектурой компьютера называется ее логическая организация, структура и ресурсы, которые может использовать программист. Практически все модели современных ПК имеют *магистральный* тип архитектуры.

При сборке компьютера используется *принцип открытой архитектуры*, т.е. компьютер состоит из независимо изготовленных частей, и принципы конструкции доступны всем желающим. Таким образом, пользователи имеют возможность самостоятельно модернизировать свои компьютеры и оснащать их дополнительными устройствами различных производителей.

3.2.Системный блок

Системный блок реализует все основные процессы по переработке информации, осуществляя оперативное хранение данных, и управляет работой всех блоков, обеспечивая их системное взаимодействие. Возможности компьютера во многом определяются компонентами системного блока.

Системный блок ПК состоит из таких основных блоков, как основная память, процессор, периферийные устройства. Средством объединения составляющих компьютера в единую систему является шина. Она передает адреса памяти, данные и управляющие сигналы.

Системная шина предназначена для организации взаимодействия периферийных устройств с ядром компьютера, в состав которого входят микропроцессор, ОЗУ и ПЗУ. Чем выше скорость системной шины, тем выше скорость обмена информацией с периферийными устройствами.

Локальная шина обеспечивает связь микропроцессора с быстродействующей периферией (ОЗУ, ПЗУ, внешней памятью и др.).

Процессор и основная память находятся на большой плате, которая называется *материнской*. Для подключения к ней периферийных устройств (дисководов, манипуляторов типа мыши, принтеров и т.д.) служат специальные платы - контроллеры. Они вставляются в разъемы (слоты) на материнской плате, а к их концу (порту), выходящему наружу компьютера, подключается дополнительное устройство. Таким образом, периферийные устройства подключаются к системной магистрали не непосредственно, а через специальные устройства - контроллеры.

Микропроцессор (CPU – Central Processing Unit) – центральный процессор, выполняющий вычисления в соответствии с заданной программой и управляющий работой компьютера. В ранних поколениях ПК использовались два процессора: основной – для выполнения логических и математических операций, и математический сопроцессор – для ускорения выполнения вычислений над числами с плавающей точкой. В современных ПК эти функции выполняет микропроцессор. Кроме микропроцессора, на материнской плате размещаются:

- *оперативное запоминающее устройство* (ОЗУ, RAM – Random Access Memory) – память произвольного доступа для размещения программ и данных, которые выполняются компьютером в данное время;
- *постоянное запоминающее устройство* (ПЗУ) – память, которая содержит базовую систему ввода-вывода (BIOS), управляющие программы и команды.

Основными характеристиками микропроцессора являются его *тактовая частота* и *разрядность*, которые определяют быстродействие компьютера. *Такт* – это время, за которое процессор выполняет элементарную операцию по обработке информации.

Разрядность определяет, сколько байт информации может одновременно обрабатываться за 1 такт, каким образом осуществляется адресация оперативной памяти и ее объем, обращение к другим запоминающим устройствам. В настоящее время наиболее широко используются 32-разрядные процессоры.

IBM–подобные компьютеры используют микропроцессоры фирмы Intel. 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.).
- компьютеры на базе процессора Intel Itanium™ (2001).
- компьютеры на базе процессоров Intel Xeon™ MP и Intel® Itanium 2 (2002).

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

На рынке компьютеров представлены микропроцессоры и других производителей, таких как AMD (K7 (Athlon), Thunderbird, SledgeHamme) и Cyrix (6x86 или MI, 6x86MX и MII).

Программы и данные, определяющие работу компьютера после включения питания, хранятся в *постоянном запоминающем устройстве* (ПЗУ).

В *оперативном запоминающем устройстве* (ОЗУ), или оперативной памяти, хранятся данные и программы, участвующие в текущий момент в решении задачи и организации работы компьютера. Вся информация, которая находится в ОЗУ, после выключения компьютера исчезает. Быстродействие ОЗУ меньше быстродействия микропроцессора. Для повышения производительности вводят *кэш-память*, которая построена на более быстродействующих по сравнению с ОЗУ микросхемах. Она позволяет уменьшить время доступа к памяти. В ней хранятся наиболее часто используемые данные, и она является промежуточной между микропроцессором и ОЗУ [8]. Чем больше эта память, тем выше может быть производительность компьютера. Для работы с современным программным обеспечением емкость оперативной памяти компьютера должна быть не менее 32 Мб.

Процесс взаимодействия процессора и памяти сводится в основном к двум операциям: запись информации в память и чтение информации из памяти. При записи процессор по специальным проводникам (шине адреса) передает биты, кодирующие адрес; по другим проводникам (шине управления) передает управляющий сигнал – «запись», по третьей группе проводников (шине данных) – передает записываемую информацию. При чтении также по шине адреса передается соответствующий адрес оперативной памяти и с шины данных считывается требуемая информация.

Число одновременно передаваемых по шине адреса и шине данных разрядов (битов) называется разрядностью соответствующей шины и является важной характеристикой ЭВМ. Разрядность шины данных определяет максимальное общее количество доступной памяти (адресное пространство процессора); разрядность шины данных - максимальную порцию информации, которую можно получить из памяти за один раз. Для процессора i8088, на котором работал первый персональный компьютер корпорации IBM, разрядность адресной шины равнялась 20 и, соответственно, максимальное количество доступной памяти равнялось 1 Мбайту. Современные процессоры Pentium могут адресовать до 1 Гбайта памяти.

Кроме того, в состав системного блока ПК входят:

- *устройства управления* (контроллеры и адаптеры), управляющие работой различных устройств, входящих в состав компьютера;
- *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);
- *порты ввода-вывода*, через которые производится обмен данными с внешними устройствами.

Накопители на жестких магнитных дисках (НЖМД, HDD – Hard Disk Drive), или *винчестеры* – предназначены для длительного хранения информации. Жестких дисков может быть несколько. Скорость вращения диска в типовых моделях составляет 5400-7200 об/мин. Информация, хранимая на этих дисках, практически не подвержена влиянию механических воздействий. Емкость и производительность винчестера являются сейчас важнейшими характеристиками компьютера в целом.

Обмен данными с внутренними устройствами компьютера осуществляется через *специализированные* порты. К портам *общего назначения* могут подсоединяться дополнительные внешние устройства (принтер, мышь, и т.п.).

Различают следующие виды портов [11]:

параллельный порт – обозначается LP1, LP2 и т.д. (LP – Line Printer). Через параллельные порты с компьютером соединяются принтеры и другие устройства, требующие высокую скорость передачи данных, а также их можно использовать для соединения двух компьютеров между собой;

последовательный порт (Serial port или COM-port (Communications port)) – это порт, через который данные передаются только в одном направлении в каждый момент времени. На схемах эти порты обозначают COM1, COM2 и т.д.

USB (Universal Serial Bus) – универсальный последовательный порт. Этот порт позволяет подключать различные периферийные устройства. В настоящее время производители периферийных устройств выпускают их в двух вариантах – с обычными для этих устройств портами (разными для разных устройств) и USB-портами. Важной особенностью USB портов является то, что они поддерживают технологию *Plug and Play,* т.е. при подключении устройства не требуется устанавливать драйвер для него. Кроме того, порты USB поддерживают

возможность подключения к ним при работающем компьютере («горячее подключение»).

Порты PS/2 – это параллельные порты для мыши и клавиатуры.

AT/MIDI порт (Musical Instrument Digital Interface – соединение с цифровыми музыкальными инструментами) – это порт, через который первоначально (до PS/2) подключалась клавиатура, а в настоящее время, в основном, подключаются музыкальные клавиатуры и синтезаторы.

FireWire (дословно «огненный провод») – последовательный порт, поддерживающий скорость передачи данных в 400 Мбит/сек. Он служит для подключения к компьютеру видеоустройств, таких как, например, видеомагнитофон, а также других устройств, требующих быстрой передачи большого объема информации, например, внешних жестких дисков. Порты FireWire, как и порты USB, поддерживают технологию Plug and Play и «горячего подключения» [11].

Внешние запоминающие устройства (ВЗУ) обеспечивают длительное хранение информации на различных носителях.

Накопители на гибких магнитных дисках (НГМД, FDD – Floppy Drive Disk) являются устройством со сменным носителем информации.

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 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 (привода) и записываемого компакт-диска. Диски *CD-R* бывают двух объемов 650 Мб и 700 Мб.

CD-RW (Compact Disk ReWritable - компакт диск перезаписываемый) — это оптическая система многоразовой записи. Она состоит из дисковода и пере-

записываемого компакт-диска. Диски *CD-RW* также бывают двух объемов 650 Мб и 700 Мб.

DVD (Digital Versatile Disc) - цифровой универсальный диск. Диск DVD внешне не отличается от диска CD-R или *CD-RW*, но для записи на него используется другая технология. В DVD в несколько раз увеличена плотность записи и уменьшены размеры наносимых меток.

DVD-диски бывают трех типов:

- односторонний однослойный. Запись производится на один слой. Диск вмещает 4,7 Гб информации;
- односторонний двухслойный. Запись производится на два слоя. Верхний слой является полупрозрачным. Диск вмещает 8,5 Гб информации;
- двухсторонний двухслойный. Запись производится на два слоя, с двух сторон. Диск вмещает 17 Гб информации. Дисковод для таких дисков имеет два лазера - сверху и снизу.

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

3.3. Периферийные устройства персональных компьютеров

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. Клавиатуры разных типов компьютеров различаются количеством клавиш и их обозначением. 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 и имеет не менее 2 клавиш. Мышью выполняют такие операции управления компьютером, как *перемещение* – мышь перемещается по коврику без нажатия клавиш, чтобы курсор занял на экране нужное место; *щелчок* (click) – однократное нажатие клавиши мыши; *двойной щелчок* (double-click) – двукратное быстрое нажатие клавиши мыши; *протяжка* (drag) – перемещение мыши с нажатой клавишей.

Монитор (дисплей) – это основное устройство, служащее для отображения выводимой на экран информации. Он состоит из двух основных частей – экрана и электронного блока, размещенных в одном корпусе. По функциональным возможностям дисплеи подразделяются на *алфавитно-цифровые,* способные воспроизводить ограниченный набор символов, и *графические*, позволяющие отображать графическую и символьную информацию.

По физическим принципам формирования изображения можно выделить мониторы на *базе электронно-лучевой* трубки – ЭЛТ-мониторы (CRT- Cathode Ray Tube). Так как в электронно-лучевой трубке электронно-лучевая пушка

расположена перпендикулярно плоскости экрана, монитор имеет довольно большой объем и занимает много места на рабочем столе. Поэтому в настоящее время ЭЛТ-мониторы вытесняются *жидкокристаллическими* дисплеями (LCD Liquid Crystal Display).

В таких мониторах используется свойство жидких кристаллов упорядочиваться под действием электрического поля и изменять при этом угол поляризации проходящего через кристаллы света. Угол поляризации света изменяется в зависимости от величины электрического поля. Панель монитора состоит из двух прозрачных пластин, между которыми находятся жидкие кристаллы. Снаружи на эти пластины нанесены тонкие параллельные прозрачные электроды. На одной пластине они располагаются вертикально, на другой – горизонтально, что создает объемную координатную сетку. Если пропустить ток по одному вертикальному и горизонтальному проводникам, то в ячейке, находящейся между ними, возникает электрическое поле, воздействующее на кристаллы.

По сравнению с ЭЛТ-мониторами ЖК-мониторы безопасны, занимают мало места и потребляют меньше энергии. Жидкокристаллические мониторы широко используются в переносных компьютерах, наладонниках, а в последнее время – и в настольных компьютерах.

Возможности компьютера по отображению информации определяются совокупностью и совместимостью технических характеристик дисплея и его адаптера. 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. Разрешение монитора измеряется количеством *пикселов* (точек), высвечиваемых по горизонтали (m) и вертикали (n), и указывается в виде произведения mxn. Чем больше пикселов может быть воспроизведено на экране одного и того же размера, тем выше качество изображения.

В настоящее время повсеместно используются видеоадаптеры типа Super VGA (SVGA) 32-х и 64-х битные с разрешением 1024x768 точек и выше: 1152x864, 1280x1024. В некоторых сферах деятельности, например, в полиграфии или обработке изображений, требуются специальные мониторы.

Монитор в зависимости от адаптера, которым он комплектуется, может быть монохромным цифровым, монохромным графическим или цветным графическим.

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

Основным узлом *матричного принтера* (dot matrix) 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. В матричных принтерах каждый символ или фрагмент графического изображения воспроизводится совокупностью точек. Лента заправляется в принтер в специальных пластмассовых блоках – картриджах.

Скорость работы матричных принтеров зависит от операционной системы и текстового редактора, установленных на компьютере. Так скорость матричного принтера при работе с приложениями MS-DOS высокая, в то время как при печати документа из текстового редактора MS Word – достаточно низкая.

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 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 *фотодиодных принтерах* вместо лазера имеется полоса, состоящая из большого количества фотодиодов (полупроводниковых элементов, которые светятся, если к ним подвести электрический ток), свет которых электризует полупроводниковый барабан, остальные действия выполняются так же как и в лазерном принтере.

С появлением цифровых фотоаппаратов возникла необходимость использовать принтеры для печати обычных бумажных фотографий. Для этой цели были разработаны *сублимационные* принтеры. Сублимационная технология печати ранее использовалась в цветных копировальных аппаратах.

В таких 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 происходит на бумаге, по составу похожей на обычную фотобумагу, но без желатинового слоя. Бумага для фотопринтеров бывает матовой и глянцевой.

К *устройствам ввода-вывода информации*, имеющим специфическое назначение, можно отнести [11]:

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, которое с помощью оптики считывает графическую и текстовую информацию и вводит ее в компьютер.

Модем (Modem, модулятор-демодулятор) – устройство для передачи сигнала из компьютера через телефонную линию на другой компьютер или на телефон. Через модем компьютер можно соединить с веб-сервером и получить доступ в Интернет. Модем преобразует аналоговый телефонный сигнал в цифровой компьютерный и наоборот. Основной характеристикой модема является его скорость работы или скорость передачи данных. Для работы в Интернет рекомендуются модемы со скоростью не менее 14400 бит в секунду. Модемы бывают *внутренними* (плата, вставляемая в гнездо расширения) и *внешними*, подключаемыми к одному из портов компьютера.

Звуковая система – это комплект оборудования, состоящий из звуковой платы, стереоколонок или наушников. Она служит для воспроизведения и записи качественного стереозвука.

Звуковая плата – это электронная плата, на которой располагается стереоусилитель, аналого-цифровой и цифро-аналоговый преобразователи, цифровой сигнальный и аудио- процессоры, FM-синтезатор, волновой синтезатор, входной и выходной микшеры, темброблок, аудиопорты, порты для микрофона, наушников или звуковых колонок, игровой порт (он же MIDI-порт).

Звуковая система позволяет использовать мультимедийные возможности компьютера. Через нее к компьютеру может подключаться магнитофон, радиоприемник, микрофон. При наличии специальных программ можно звук с магнитной ленты магнитофона перевести в цифровой формат и записать в виде звукового файла – оцифровать звук. Можно также цифровой звук из звукового файла записать в обычном аналоговом виде на магнитную ленту. Таким же образом можно оцифровывать звуки из радиоприемника.

Американская компания CREATIVE выпускает звуковые системы для компьютеров под названием Sound Blaster. Это – торговая марка компании, но в обиходе часто любую звуковую систему называют Sound Blaster.

3.4. Вопросы для самоконтроля

1. Перечислите основные аппаратные средства ПК.
2. Назовите основные компоненты системного блока.
3. Назовите основные составляющие материнской платы.
4. Чем характеризуются микропроцессоры?

5. Приведите классификацию персональных компьютеров в зависимости от установленного микропроцессора.

6. Что такое шина, и какие функции она выполняет? Охарактеризуйте виды шин.

7. Перечислите основные порты общего назначения.

8. Перечислите основные порты специализированного назначения.

9. Дайте понятие технологии Plug And Play. Какие порты поддерживают эту технологию?

10.Перечислите и охарактеризуйте современные внешние запоминающие устройства.

11.Перечислите и охарактеризуйте основные устройства ввода-вывода информации.

12.По каким признакам классифицируются мониторы?

13.Какие виды мониторов распространены в настоящее время?

14.Объясните физические принципы формирования изображения в жидкокристаллических мониторах.

15.Что такое видеоадаптер? Назовите основные виды видеоадаптеров.

16.Какие устройства предназначены для вывода алфавитно-цифровой или графической информации на бумагу? Перечислите их виды.

17.Какой принцип печати заложен в основу работы матричного принтера?

18.Что лежит в основе работы лазерного принтера?

19.На чем основана работа струйного принтера?

20.Перечислите и охарактеризуйте дополнительные устройства ввода-вывода информации.

ГЛАВА 4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

4.1. Основные понятия, функции и составные части операционной системы

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

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

Операционная система – это совокупность программ, содержащая основные инструкции для центрального процессора по управлению компьютером и всеми его элементами [7]. Операционные системы могут классифицироваться по следующим признакам:

- число пользователей, одновременно работающих с системой: однопользовательские и многопользовательские;
- число задач, которые могут решаться с их помощью в любой момент времени: однозадачные и многозадачные;
- основной способ общения пользователя с ОС: диалог на языке команд, диалог на языке меню, диалог на языке графических представлений;
- число разрядов адресной шины: 16, 32 или 64;
- минимально необходимые объемы оперативной и дисковой памяти, класс микропроцессора.

Являясь неотъемлемой частью компьютера, операционная система обеспечивает поддержку работы программ с аппаратурой и предоставление пользователю возможностей общего управления компьютером. В настоящее время существует множество операционных систем, 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 операционных систем 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 операционной системы, например, поддержку СУБД, графический интерфейс, и т.д. Ядро операционной системы постоянно находится в памяти. Остальное программное обеспечение располагается на внешних запоминающих устройствах.

Выбор операционной системы осуществляется с учетом таких характеристик, как распространенность, наличие большого количества прикладных программных средств, работающих под ее управлением, простота освоения и взаимодействия с ней пользователя, легкость перехода с одной версии операционной системы к другой, более совершенной.

Основными препятствиями к использованию прикладного программного обеспечения в различных операционных системах являются разные способы организации файлов и несовместимость форматов дисков (различное число секторов и различная емкость).

В различных компьютерах 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.

4.1.1. Операционная система MS DOS

Ранее в IBM-совместимых компьютерах наиболее часто использовалась операционная система MS DOS, разработанная компанией Microsoft. Одним из назначений MS DOS была работа с файловой системой.

MS DOS (Microsoft Disk Operation System) – это 16-разрядная, однопользовательская, однозадачная ОС, общающаяся с пользователем на языке команд. Она принята в качестве базовой для работы на компьютерах IBM PC. Программы MS DOS работают и выполняются в пределах одного мегабайта адресного пространства компьютера, а основная память может использоваться для хранения данных. Основными достоинствами MS DOS является наличие развитого командного языка, а также возможностей организации многоуровневых каталогов, работы со всеми пользовательскими устройствами, как с файлами и подключения пользователем дополнительных внешних устройств.

Для MS DOS разработан большой арсенал программных средств. Имеется несколько вариантов трансляторов практически для всех популярных алгоритмических языков высокого уровня, таких, как Basic, Pascal, C, Fortran, Кобол, Лисп, Пролог и др., 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 году, последняя версия MS DOS 6.2 (или русский вариант 6.22) – в 1994 году.

Важнейшей особенностью MS DOS является ее *модульность*, что позволяет при необходимости расширения функций системы модифицировать ее модули.

В состав ОС MS DOS входят следующие модули:

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

2. *Системный загрузчик* размещается в первом секторе нулевой дорожки системного диска. Он считывает с диска в оперативную память модуль расширения BIOS и базовый модуль.

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

4. *Командный процессор* или интерпретатор команд (файл COMMAND.COM) принимает, анализирует и выполняет команды пользователя и обрабатывает командные файлы (файлы типа BAT).

5. *Утилиты MS DOS* – программы обслуживающего характера, поставляемые вместе с операционной системой в виде отдельных файлов с расширением COM или EXE. Например, утилита для форматирования дискет и т.п.

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

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, например, COMMAND.COM.

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 изменить, 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.

Файловая система MS DOS имеет *иерархическую* (древовидную) структуру.

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 регистрируют-

ся другие файлы и каталоги. Так образуется разветвленная (древовидная) файловая структура на диске (рис..4.1).

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, Primer1.pas и т.д.).

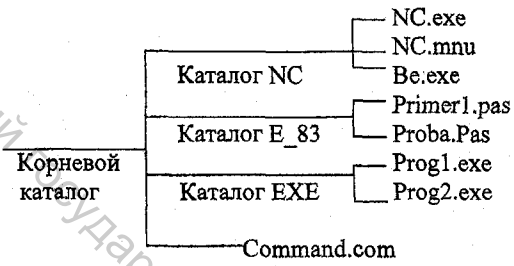


Рис.4.1. Дерево каталогов

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, `h:\E_83\Proba.pas.`

Для MS DOS создано множество прикладных программ, которые могут работать на различных компьютерах, 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. Наиболее известные такие программные надстройки как Norton Commander, PCToolls, Volkov Commander и некоторые другие. Они дают возможность компактно размещать на накопителе магнитных дисков необходимые 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 операционных систем связано с поддержкой новых технических средств, в частности, с расширением до нескольких десятков и более мегабайт оперативной памяти, новых видов накопителей, ви-

дисмониторов и т.д. Так Windows 3.1, разработанная в 1991 г., и позже Windows 3.11, как сетевой вариант, явилась операционной оболочкой, запускаемой на выполнение как обычная программа MS DOS.

4.1.2. Семейство операционных систем WINDOWS

Операционная система *Windows 3.1* представляет собой однопользовательскую многозадачную операционную систему, работающую на базе MS DOS и использующую внутренние функции и процедуры этой операционной системы. Принципиальным отличием программных приложений, предназначенных для работы в среде Windows 3.1, является то, что они работают 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 драйверу, непосредственно отвечающему 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 графически визуализированного элемента (кнопки, пиктограммы, диалогового окна, мастера подсказок и др.) с помощью мыши, а команды клавиатуры и «горячие клавиши» имеют резервное значение. Сетевая версия этой оболочки представлена Windows 3.11 [2].

Операционная система *Windows 95* представляет собой высокопроизводительную, многозадачную, 32-разрядную операционную систему с графическим интерфейсом и расширенными сетевыми возможностями, работающую в защищенном режиме, поддерживающую 16-разрядные приложения без всякой их модификации. Windows 95 уже не требовала поддержки MS DOS. Эта интегрированная среда обеспечивает эффективный обмен текстовой, графической, звуковой и видеоинформацией между различными программами за счет возможности использования CD-проигрывателя и поддержки видеодисков. Она полностью совместима с используемыми в настоящее время аппаратными и программными средствами.

Windows 95 использует технологию Plug and Play, упрощая работу с компьютером за счет помощи при распознавании устройств для их установки и настройки, интеграции драйверов устройств системных компонентов и пользовательского интерфейса, возможности динамического изменения конфигурации системы, упрощения настройки устройств и управления оборудованием. Специальная программа *диспетчер устройств* позволяет получать информацию обо всех найденных системой устройствах и изменять при необходимости их конфигурацию.

Операционная система *Windows 98* по сравнению с Windows 95 включает программные средства, позволяющие компьютеру работать быстрее без добав-

ления нового оборудования, имеет повышенную надежность за счет использования новых возможностей и мастеров.

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 неправильного выключения компьютера. Она обнаруживает наиболее вероятные повреждения дисков и исправляет ошибки. Пользователь имеет возможность выполнить проверку диска в любой момент.

Проверка реестра позволяет обслуживать и проверять реестр на наличие ошибок и несогласованности структур данных при каждом запуске ПК.

Программы архивации предоставляют расширенные возможности архивации и восстановления данных.

Мастер обслуживания позволяет быстрее выполнять программы, проверять жесткий диск на наличие ошибок и освобождать место на диске.

Обозреватель Интернета *Internet Explorer* делает доступными с рабочего стола такие возможности как поиск в сети; визуализация канала Web-узлов на рабочем столе; подписка на избранные узлы; настройка панели ссылок и панели обозревателя; контроль содержимого из зоны безопасности при поиске Web. Это приложение объединяет рабочий стол с Web, благодаря чему рабочий стол и его папки выглядят и действуют также, как и при работе с сетью. Кроме того, на рабочий стол и на панель задач можно добавить активное содержимое, например, бегущую строку новостей.

Windows NT 4.0 – это полная 32-х разрядная версия, позволяющая раздельную работу с выделением для каждого пользователя дискового пространства, защищенного паролем. Эта операционная система предназначена для работы в сети и имеет две модификации: *Windows NT Workstation* – для рабочих станций, и *Windows NT Server* – для выделенного сервера. В данных операционных системах предусмотрены дополнительные средства поддержки эффективной работы в локальной сети, улучшенные средства защиты данных от несанкционированного просмотра и случайного повреждения. Windows NT Server предоставляет расширенные средства для работы в Интернет. Эта операционная система существует только для аппаратных платформ Alpha и Intel. Широкого распространения она не получила и в настоящее время используется ограниченно.

Операционная система *Windows 2000* представлена такими модификациями как *Windows 2000 Professional* и *Windows 2000 Server*. Windows 2000 Professional разработана с учетом потребностей пользователей Интернет, оснащена дополнительными возможностями для мобильных пользователей и используется в настольных и переносных компьютерах. Она сохранила все полезные возможности Windows 98 и дополнена системой безопасности, средствами управления и обеспечения надежности, характерными для Windows NT.

Серверные приложения Windows 2000 Server представлены такими версиями, как *Windows 2000 Server*, используемой для обеспечения безопасной работы в сети

Интернет, *Windows 2000 Advanced Server*, ориентированной на использование бизнес-приложений и поддержку большего числа пользователей и более сложных приложений, чем Windows 2000 Sever, и *Windows 2000 Datacenter Server*, имеющей в дополнение к Windows 2000 Advanced Server расширенные возможности и более высокую производительность работы с памятью.

Windows 2000 по сравнению с ранее разработанными версиями имеет ряд новых технических возможностей:

1. Защита данных средствами протокола *аутентификации*, обеспечивающего передачу данных незащищенным путем.

2. Инфраструктура открытых ключей для защиты информации. Она представляет собой систему цифровых сертификатов и организаций, удостоверяющих сертификаты CAS и дает возможность обоим участникам транзакции (участникам обмена сообщениями) проверить, действительно ли партнер является тем, за кого себя выдает, а также шифровать транзакции. Такая система защиты данных лучше всего подходит для использования в сети Интернет и является полезным средством для организации электронных конференций между предпринимателями.

3. Наличие *маршрутизатора*, обладающего интерфейсом подключения по запросу, облегчает процесс установки соединения с интернет-провайдером. При помощи *сервер-доступа* 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 сеть, выходить через маршрутизатор в Интернет, используя единственный IP-адрес от Internet-провайдера. Таким образом, множество пользователей локальной сети являются, с точки зрения провайдера, единым клиентом.

4. Встроенные средства удаленного доступа.

5. Новые функции управления хранением данных: управление квотами дискового пространства, иерархическую систему хранения данных, динамическое управление ими, а также внесение 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 «засыпания» (sleep). Перед отключением питания все содержимое физической памяти записывается в специальный файл.

7. Расширенное административное управление, позволяющее снизить общую стоимость эксплуатации ОС.

Операционная система *Windows XP* официально была представлена 25 октября 2001 года. Она заменила Windows 2000, которая так и не приобрела 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.

Windows XP Home Edition предназначена для домашнего использования и малого бизнеса, и является модификацией Windows 9x.

Windows XP Professional ориентирована на применение в сфере бизнеса. Хотя XP Home Edition и XP Professional базируются на одном и том же ядре, ОС XP Professional обладает более расширенной функциональностью по сравнению с домашней системой.

Windows XP построена на усовершенствованном ядре Windows 2000 и имеет новый, простой интерфейс, который упрощает работу с компьютером. Служба терминалов Remote Assistance позволяет дистанционно подключаться к компьютеру. Таким образом, можно получить доступ к любым документам и файлам на удаленной машине. Откат драйвера (Rollback Driver) препятствует сбою системы. В Windows XP добавлена функция «System Restore» или 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.

Windows XP позволяет записывать компакт-диски без привлечения посторонних программ. Также имеется встроенный архиватор типа Zip. Обновлен обозреватель интернета Internet Explorer (версия 6.0), а также почтовая программа Outlook Express 6. Брандмауэр интернет-подключений автоматически защищает компьютер, подключенный к интернету, от несанкционированного доступа. Проигрыватель Windows Media для Windows XP обеспечивает поиск, воспроизведение, упорядочивание и хранение цифрового мультимедиаматериала. Служба сообщений Windows Messenger является эффективным средством связи и поддерживает передачу немедленных сообщений, проведение голосовых и видеоконференций, а также совместное использование приложений.

В Windows XP реализовано беспроводное подключение — автоматическая беспроводная конфигурация сети с использованием стандарта 802.1x, а также автономные файлы и папки — доступ к файлам и папкам, хранящимся на общем сетевом диске возможен даже во время отключения компьютера от сервера, шифрованная файловая система — защита важных данных, содержащихся в файлах, хранящихся на диске, на котором используется файловая система NTFS, и управление доступом — запрещение доступа к избранным файлам, приложениям или другим ресурсам.

Технология Single Worldwide Binary для отображения текста позволяет вводить текст на любом языке и запускать версию приложений Win32 для любого языка, используя соответствующую версию ОС Windows XP.

Многоязычный пользовательский интерфейс позволяет менять язык пользовательского интерфейса для работы с локализованными диалоговыми окнами, меню, файлами справки, словарями и т.д.

Подводя итог, можно выделить следующие общие характеристики Windows:

- Windows является интегрированной операционной системой, ядро которой загружается в момент включения компьютера, активизируя графический интерфейс и обеспечивая полную совместимость с ОС MS DOS.
- Вытесняющая многозадачность – возможность ОС самостоятельно в зависимости от внутренней ситуации передавать или забирать управление того или иного приложения, не позволяя одному приложению забирать все аппаратные ресурсы, позволяет использовать несколько приложений одновременно.
- Windows является многопоточной операционной системой, т.е. способной выполнять операции одновременно над потоками нескольких 32-битных приложений, называемых процессами.
- Имеются широкие возможности для пользователей с ограниченными возможностями, такие как возможность масштабирования элементов интерфейса, запоминающие клавиши, режим Mouse Keys (все действия с мышью можно выполнить через клавиатуру) и др.

4.1.3. Семейство операционных систем Unix

Семейство операционных систем Unix является альтернативой семейству Windows. Ее основным отличием и преимуществом является возможность использования 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 многозадачной и многопользовательской системой. 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.

Одной из наиболее популярных версий Unix для персональных компьютеров является ОС Linux. Ее разработка была начата в 1991 г. студентом из Хельсинки Линусом Торвальдсом. Она распространяется с исходными текстами программ (с открытым кодом), поэтому доступна для модернизации. Ее особенностями являются поддержка национальной и стандартной клавиатур, наличие динамически загружаемых драйверов, поддержка различных версий файловых систем и полного набора протоколов TCP/IP для работы в сети, реализация файловой системы MS DOS, позволяющей прямо обращаться к файлам MS DOS на жестком диске.

4.2. 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.

4.2.1. Программы обслуживания диска

Форматирование диска. Накопители на магнитных носителях, перед тем как их будут использовать в качестве носителей информации, должны пройти форматирование. Для этого используется стандартная программа, 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 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 кластеров, 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 только системной области или области данных. При нахождении поврежденных секторов их данные обычно переносятся в другие сектора. Некоторые системные файлы нельзя переносить, поэтому рекомендуется отменить исправление ошибок в секторах, где расположены такие файлы. 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. Если для создания новых файлов необходимо освободить место, занимаемое уже ненужными файлами, то их удаляют. Удаленные файлы помещаются в корзину на рабочем столе, т.е. они меняют место расположения, но не исчезают с диска. Для освобождения места, занятого файлами, необходимо очистить корзину.

4.2.2. 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 извлекать из этого файла исходные данные, а также добавлять новые. Наиболее популярными программами архивации для Windows, являются WinZip, WinRar, WinAzj.

4.2.3. Компьютерные вирусы и антивирусные средства

Вирус – это, как правило, небольшая по объему, скрытая до определенного момента последовательность программных кодов, характеризующаяся наличием отрицательных последствий от ее функционирования, имеющая возможность создания своих копий и внедрения их в другие программные объекты и производящая несанкционированные со стороны пользователя действия.

Причины создания и развития индустрии вирусов могут иметь технический (ранние версии операционных систем), экономический (борьба с конкурентами) и субъективный (компьютерное хулиганство) характер.

Вирусы классифицируются как загрузочные, файловые, макровирусы и сетевые вирусы.

Загрузочные вирусы внедряются в загрузочные области носителя. 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. Эти вирусы являются резидентными, т.е. при загрузке системы вирус устанавливается в память компьютера и находится там постоянно, перехватывая необходимое прерывание и производя все запрограммированные действия.

Файловый вирус использует особенности организации системы и внедряется 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 операционной системы, определяющее порядок запуска программы. Так, если в одном каталоге есть файлы с одинаковыми именами, то сначала запускается файл с расширением *.BAT, потом *.COM, а затем *.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 (в основном это макровирусы) 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 большого распространения.

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

Распознать вирусы достаточно сложно, так как не все вирусы обладают всеми свойствами, а также существуют программы, которые имеют подобные свойства, но не являются вирусами. Одним из основных последствий деятельности вирусов является потеря или порча информации, поэтому для обеспечения устойчивой и надежной работы желательно всегда иметь эталонные копии используемой информации и программного обеспечения. 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 позволяет не только определить наличие вирусов, но и оперативно их обезвредить и восстановить испорченную информацию. Популярными антивирусными эффективными программными средствами являются антивирус Касперского (AVP), Norton Antivirus, Doctor Web (DRW).

К сожалению, не существует антивирусов, которые бы обеспечили полную защиту от вирусов. При выборе антивирусного обеспечения следует при-

нимать во внимание перечень и качество предоставляемых услуг, таких как 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 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.

4.3. Системы программирования

Система программирования состоит из языка программирования, компилятора или интерпретатора программ, написанных на этом языке, соответствующей документации и вспомогательных средств для подготовки программ к выполнению. *Программа-компилятор* служит для преобразования программы с языка программирования на машинный язык. *Программа-интерпретатор* осуществляет покомандную обработку и выполнение исходной программы.

Машинно-ориентированные языки – это языки, более понятные компьютеру. Среди наиболее распространенных машинно-ориентированных языков можно выделить Ассемблер. Между тем, 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 непрограммист.

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

К первой группе можно отнести *алгоритмические языки,* которые позволяют заложить в программу алгоритм решения. Наиболее известными представителями являются Фортран, Алгол, Ада, Бейсик, Паскаль, Модула-2, Delphi, Си.

Фортран – формульный транслятор, предназначенный для решения научно-технических задач. Разработан фирмой IBM в середине 50-х годов. Недостатком Фортрана является то, что он плохо поддается структурированию, и 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 позволяет обрабатывать не только числовые, но и символьные переменные, поэтому удобен для решения примитивных повседневных задач. Разновидности: TurboBasic, GWBasic, Visual Basic for Applications.

Паскаль (назван так в честь знаменитого математика Блеза Паскаля) является языком 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, ПЛ/1 и 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. Привлекательная черта C и C++ заключается в наличии 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 Бейсика или Паскаля, является языком функционального программирования, ориентированным на динамическую обработку данных. Эффективен для работы со списками, может быть использован для задач, где отсутствует четкий алгоритм решения.

Пролог – программирование в логике. Европейский 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.

С появлением среды MS Windows появилась *третья* группа языков программирования, при помощи которых можно создавать Windows-совместимые программы – Visual Basic, Visual C и др.

Сравнительно молодым языком программирования является язык Java, созданный фирмой Sun Microsystems в 1995 году. Этот язык позволяет создавать обычные программы и программы для использования в Интернет. Для создания программ многие фирмы создали оболочки, позволяющие разрабатывать программ на Java: Sun – Java WorkShop; Microsoft – Visual J++; Borland – Jduilder; Symantec – Visual Cafe; IBM – Visual Age и др.

Четвертую группу составляют *языки для написания web-страниц*. К ним относятся *язык разметки web-страниц HTML* Hyper Text Markup Language) и *скриптовые языки.*

Скриптовые языки – специальные языки программирования, которые содержат инструкции для браузера и web-сервера. Эти инструкции называются *скриптами*. Скриптовые языки делятся на два класса – *серверные* скриптовые языки и *клиентские* языки. Серверные скрипты выполняются web-сервером, клиентские – браузером клиента. Серверные языки для web-сервера операционной системы Unix – это PHP и Perl, для web-сервера операционной системы Windows – это язык ASP. Клиентские языки – Java Script, VisualBasic Script.

4.4. 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, QuarkXPress, MS Publisher;
- *графические редакторы* – пакеты, предназначенные для обработки графической информации. Различают *растровые* графические редакторы, рисующие изображение по точкам разных цветов: Aldus Photo, Adobe PhotoShop, Photo Finish, 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: Power Point, World Perfect Presentation, Frulance Graphics;
- *пакеты программ мультимедиа* – предназначены для использования IBM 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;
- *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.

4.5. 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 XP.
12. Какие новые технические решения использованы в ОС Windows 2000?
13. Перечислите и поясните различия между ОС 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?

ГЛАВА 5. ТЕКСТОВЫЕ ПРОЦЕССОРЫ

5. 1. Классификация текстовых процессоров

Текстовый процессор – это специальная программа для создания и обработки текстов, которая позволяет подготовить, отформатировать и распечатать текстовые документы различной длины и структуры.

Текстовые процессоры можно разделить *по типу обрабатываемого объекта* на четыре группы [8]:

- *редакторы текстов* – предназначены для создания и редактирования простых текстов или программ: MultiEdit, Norton Editor;
- *редакторы научных текстов* – обеспечивают подготовку и редактирование научных текстов, содержащих большое количество математических формул, графиков, специальных символов и т.п.: MathOr, Tex;
- *редакторы документов* – предназначены для работы со сложными документами, которые могут содержать вложенные документы, страницы, абзацы, таблицы, диаграммы, рисунки, которые могут быть созданы и в других приложениях: Лексикон, AmiPro, Microsoft Word. Современные редакторы документов имеют возможности подготовки документов с формулами;
- *издательские системы* – используются для подготовки больших, сложных документов (книги, альбомы, журналы, газеты), а также для верстки текста: Corel Ventura Publisher, Adobe PageMaker, QuarkXPress. Верстка текста заключается в размещении текста по страницам создаваемого документа, вставке рисунков, использовании различных шрифтов применительно к документам, уже созданным с помощью редакторов документов. Эти системы предполагают применение редакторов документов для предварительной подготовки материалов, а для окончательной верстки используется уже сама издательская система. Многие функциональные возможности издательских систем имеют и современные редакторы документов.

Следует отметить, что отличия между процессорами различных типов перестали быть принципиальными.

5. 2. Основные возможности текстового процессора MS Word

Текстовый процессор MS Word является самым популярным редактором документов и используется для подготовки текстовой документации различного уровня сложности. Вводимый документ отображается на экране в том виде, в каком он будет выведен на печать. Пользователю доступны такие средства редактирования, как изменение типа и начертания шрифта, установка и выравнивание абзацев, установка межстрочных интервалов, выделение рамками,

многоколоночный текст, выделение абзацев, висячие отступы, автонумерация страниц и многое другое. Созданный документ можно сохранить на диске, в том числе и с указанием пароля для его защиты, или считать с диска.

В текст можно вставлять таблицы, ячейки которых могут содержать как текст, числа, графические объекты, формулы и т.д., но не могут содержать таблицы. Встроенное средство построения диаграмм позволит легко простроить графики различных типов. В таблицах можно проводить различные несложные вычисления. Текст может сопровождаться формулами, а также графикой. Графические объекты можно вставлять из уже созданных файлов различных форматов, а также создавать, используя встроенный графический редактор.

Технология IntelliSiense позволяет автоматизировать многие типовые и часто используемые операции, такие как изменение строчного символа в начале строки на прописной, динамическая коррекция ошибок, вставка автотекста. При многократном использовании одной и той же фразы автотекст позволяет указать сокращение, при появлении которого в тексте будет выполняться автоматическая замена на нужную фразу.

MS Word использует следующие типы файлов:

- **.doc* – файл основного документа;
- **.dot* – файл шаблона;
- **.wmf* – файл иллюстрации.

Интерфейс MS Word, как и любого программного приложения Windows, представляет собой набор меню и окон. При запуске MS Word автоматически создается одно пустое окно с заголовком *Документ*. Такое окно называется *окном документа*. Окно документа, с которым в данный момент работает пользователь, называется *активным*. Заголовок *Документ* с порядковым номером окна после сохранения в файл заменяется именем файла, например, Глава 5.

В правом верхнем углу находятся кнопки управления окном программы  - свернуть, восстановить обратно, закрыть. Ниже расположена строка командного меню, под которой находятся кнопки панелей инструментов – кнопок с пиктограммами, элементы которых объединены по функциональному признаку. Наиболее часто используемые команды собраны на панелях *Стандартная* и *Форматирование*. Остальные панели можно подключать, используя команду *Панели инструментов* из меню *Вид*. Кроме того, существует возможность добавления необходимых пиктограмм без подключения соответствующей панели, также используя команды меню *Вид*: *Вид* ⇒ *Панели инструментов* ⇒ *Настройка*.

Команды MS Word можно вызвать одним из 4 способов:

- *с помощью командного меню* – выбор команды осуществляется из раскрывающегося списка, организованного по многоуровневой схеме;
- *посредством пиктографического меню*;
- *из динамического меню*, появляющегося при нажатии правой кнопки мыши и содержащего команды, относящиеся к выделенному документу;

• *при помощи «горячих» клавиш* – комбинаций клавиш, позволяющих вызвать любую команду меню. Комбинации клавиш зависят от настроек операционной системы.

Все команды текстового редактора MS Word интуитивно понятны и не требуют подробных объяснений. Рассмотрим некоторые из них.

Команды меню *Файл* используются для работы с документом как с единым целым.

Создать (Ctrl+N) – позволяет создавать новый документ различного вида, в том числе с использованием шаблонов резюме, писем, факсов и т.д.

Открыть (Ctrl+O) – используется для открытия ранее сохраненного документа. Для этого в открывшемся окне следует выбрать папку, в которой находится документ.

Закрыть – закрывает окно активного документа.

Сохранить (Ctrl+S) – используется для записи изменений в файл. Если ранее имя файла не было задано, то в открывшемся окне следует указать папку, в которой будет сохранен документ, и его имя. Если файл уже имеет имя, то все изменения будут дописаны в файл с этим именем.

Сохранить как – позволит записать изменения в файл с *новым* именем. В открывшемся окне необходимо указать папку сохранения и имя файла.

Параметры страницы – используется для установки размеров полей документа, колонтитулов, размера бумаги, ориентации листа и т.д.

Предварительный просмотр – позволяет увидеть созданный документ так, как он будет выведен на печать.

Печать – используется для установки параметров печати: какие страницы необходимо печатать, в каком масштабе, сколько страниц текстового документа должно быть размещено на одном листе бумаги и т.п.

Выход – используется для корректного выхода из текстового редактора.

Команды меню *Правка* используются для работы с элементами текста.

Отменить (Ctrl+Z) – отменяет выполненное действие.

Повторить (Ctrl+Y) – повторяет отмененное действие.

Вырезать (Ctrl+X) – фрагмент текста удаляется из документа и помещается в буфер обмена.

Копировать (Ctrl+C) – фрагмент текста помещается в буфер обмена без удаления из документа.

Вставить (Ctrl+V) – вставляет фрагмент текста из буфера обмена.

Найти (Ctrl+F) – используется для поиска подстрок в тексте.

Команды меню *Вид* используются для установки и просмотра структуры документа, а также для настройки окна документа.

Разметка страницы – используется для просмотра документа с разбиением его на страницы.

Колонтитулы – используется для вставки в документ верхнего и нижнего колонтитулов, содержащих информацию, повторяющуюся вверху или внизу на всех страницах документа, а также для корректировки номеров страниц.

Команды меню *Вставка* позволяют вставлять в текстовый документ различные объекты: символы, графики, формулы, рисунки и т.д.

Разрыв – используется для принудительного разделения текста на страницы.

Номера страниц – автоматическая нумерация страниц.

Символ – вставка определенного символа.

Объект – позволяет вставлять объекты, созданные с помощью других приложений.

Рисунок – используется для вставки в документ графической информации: картинок, объектов Word Art, диаграмм, автофигур.

Команды меню *Формат* используются при форматировании документа.

Абзац – позволяет устанавливать абзацные отступы, отступы от левого и правого полей страницы, интервал между абзацами и междустрочный интервал, а также выравнивать текст относительно страницы.

Шрифт – позволяет изменять вид, цвет, размер и начертание шрифта в документе.

Список – используется для создания в документе нумерованных, маркированных и многоуровневых списков.

Границы и заливка – позволяет задавать тип, цвет и толщину линий обрамления таблиц в документе, а также вставлять различные горизонтальные линии и обрамляющие рисунки.

Команды меню *Сервис* используются для проверки правописания и внесения исправлений, настройки параметров окна. Команды меню *Таблица* используются при создании и редактировании таблиц. Основные команды этого меню будут рассмотрены далее.

В меню *Окно* отображаются все открытые окна. Активное окно отмечается значком √. Используя меню *Справка*, пользователь имеет возможность обратиться к мощной справочной системе MS Word.

5.3. Принципы создания и редактирования документа в MS Word

Создавать документ в текстовом редакторе MS Word можно одним из следующих способов: набрать исходный текст, а затем его отформатировать, либо набирать текст и последовательно его форматировать. Под *форматированием* текстового документа понимается установка параметров страницы, шрифта, абзацев, отступов, выравнивания текста, нумерации страниц и т.д.

Создание любого документа начинается с выбора из меню *Файл* с помощью команды *Создать* типа создаваемого документа – обычный документ

письмо, записка, резюме и т.д., после чего его следует сохранить. Для этого можно воспользоваться командами *Сохранить* или *Сохранить как* из меню *Файл*. В открывшемся окне следует выбрать диск, на котором будет сохранен документ, папку на диске и указать имя документа.

Далее следует установить параметры страницы, используя команду *Параметры страницы* из меню *Файл*. В открывшемся окне *Поля* необходимо указать размеры верхнего, нижнего, левого и правого полей в сантиметрах, в окне *Размер бумаги* уточнить формат и размеры листа (по умолчанию установлен формат А4 размером 21х29,7 см) и его ориентацию (книжная, альбомная). В случае необходимости изменить эти параметры. Для автоматической расстановки переносов в тексте документа из меню *Сервис* выбрать команду *Язык* ⇒ *Расстановка переносов*. В открывшемся окне поставить флажок в окне *Автоматическая расстановка переносов*. Сохранить эти изменения в уже существующем файле, используя команду *Сохранить*. После этого можно приступать к набору текста.

В окне документа имеется признак конца текста в виде горизонтальной черты, расположенной после последнего символа текста. Если окно активно, то в нем располагается *текстовый курсор* (далее – курсор) в виде вертикальной мерцающей черты. Указатель мыши в пределах текста имеет вид тонкой вертикальной черты (I), а слева от текста – направленной под углом стрелки.

Основной структурной единицей текста в Word является *абзац*, под которым понимают группу смежных строк текста, первая из которых может начинаться отступом или выступом, а все последующие начинаются с отличного от пробела символа. Концом абзаца считается строка, за которой следует либо конец документа, либо начало следующего абзаца. Конец абзаца обозначается символом ¶, который отображается на экране при нажатой пиктограмме ¶ (непечатаемые символы).

Различают следующие структурные элементы страницы: основной текст (строки, абзацы, таблицы, рисунки), верхний и нижний колонтитулы, сноски.

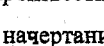
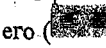
Для установки отступов можно воспользоваться меню *Формат* или использовать горизонтальную линейку. Не допускается установка абзацных отступов с помощью пробела или клавиши Tab.

Для установки отступов с помощью меню Формат необходимо вызвать команду *Абзац*. В окне *Отступы и интервалы* установить размер отступа красной строки (по умолчанию 1,25 – 1,27 см), межстрочный интервал (по умолчанию – одинарный), выравнивание текста (по ширине, по центру, правому или левому краю) и при необходимости отступ или выступ справа или слева. Отступ измеряется в положительных единицах, выступ – в отрицательных, например, 0,63 см – отступ, -0,63 см – выступ.

Используя верхнюю горизонтальную линейку, можно задавать отступ красной строки, используя нижнюю горизонтальную линейку – отступ или вы-

отступ справа или слева. Для этого установите указатель мыши на соответствующий ограничитель и протяните его на нужную величину отступа.

При работе с документом не забывать записывать изменения в файл, используя команду *Сохранить* (Ctrl+S).

Работая с текстом документа, можно изменять его фрагменты. Для этого фрагмент текста необходимо выделить, а затем произвести необходимые изменения, например выбрать вид шрифта, размер, начертание ( - полужирный, курсив, с подчеркиванием), выровнять его ( - по правому краю, по центру, по левому краю, по ширине). Фрагмент текста можно удалить из документа, выделив его и нажав клавишу <Delete>.

Фрагмент текста можно копировать из одного места в другое, оставляя его копию в исходном месте, либо перемещать (вырезать) фрагмент и вставлять его в другое место без сохранения его копии в исходном месте. Для этого используется буфер обмена.

При копировании (вырезании) необходимо выделить фрагмент текста, затем из меню *Правка* выбрать команду *Копировать* (*Вырезать*), установить курсор в том месте, где должен находиться фрагмент и вставить фрагмент, используя команду *Вставить* меню *Правка*. Все эти действия также можно выполнять, используя динамическое или пиктографическое меню, а также «горячие» клавиши.

Для написания сложных формул Microsoft Word предлагает специальную программу MS Equation 3.0 – редактор формул, вызвать которую можно, используя команду *Объект* меню *Вставка*. В открывшемся окне из предложенного списка выбрать тип объекта *Microsoft Equation 3.0*. После чего откроется окно редактора формул. Пользуясь меню, можно написать любую формулу. Аналогично можно вставлять в документ другие объекты.

Для вставки рисунков в текст используется команда *Рисунок* меню *Вставка*. После вызова этой команды откроется библиотека иллюстраций, входящих в Word. Для размещения рисунков в тексте после его вставки следует задать способ размещения рисунка на странице. Удобно это делать через динамическое меню, вызываемое правой клавишей мыши.

5.4. Создание и редактирование таблиц. Вычисления в таблицах

Word позволяет создавать и использовать таблицы для размещения в них информации различных видов – текстовой, числовой, графической и др. Каждая таблица состоит из ячеек, сгруппированных в строки и столбцы. Столбцы именуются латинскими буквами, строки нумеруются арабскими числами. Число строк и столбцов таблицы ограничено используемой версией редактора, на-

Пример, в MS Word 2000 количество столбцов – не более 63. Адрес ячейки образуется из имени столбца и номера строки, например, G5.

Выделяют такие этапы работы с таблицами, как создание таблиц, ввод и редактирование данных; форматирование таблиц; организация вычислений и сортировка.

Для создания таблицы следует воспользоваться командой *Добавить таблицу* из меню *Таблица*, указав количество столбцов и строк таблицы. Ввод и редактирование данных (текста, чисел и т.д.) осуществляется как обычно.

Для форматирования таблицы используются команды меню *Таблица*, такие как *Вставить строку* (*столбец*), *Удалить ячейки*, *Объединить ячейки* и др., а также команда *Границы и заливка* меню *Формат*, которая позволяет задавать тип, размер и цвет линий обрамления ячеек таблицы, цвет заливки и др.

В любую ячейку таблицы может быть введено не только значение, но и формула, которая может состоять из знаков операций, операндов и стандартных функций.

Формат ввода формулы

`= формула (список аргументов)`

Основные стандартные функции, поддерживаемые MS Word.

- `ABS(X)` – модуль числа;
- `AND` – логическое И;
- `AVERAGE()` – вычисляет среднее арифметическое аргументов;
- `COUNT()` – подсчитывает число ячеек в столбце от активной ячейки до конца столбца;
- `IF(X,Y,Z)` – используется для проверки логического условия и организации перехода в зависимости от его выполнения или невыполнения. `X` – логическое условие, `Y` – действие, выполняемое, если условие истинно, `Z` – действие, выполняемое, если условие ложно
- `INT(X)` – приводит значение аргумента к целочисленному типу;
- `MAX()`, `MIN()` – используется для поиска максимума или минимума среди аргументов;
- `MOD(X,Y)` – возвращает остаток от целочисленного деления аргумента `X` на `Y`;
- `NOT(X)` – отрицание (логическое НЕ);
- `OR(X,Y)` – выбор (логическое ИЛИ);
- `PRODUCT()` – вычисляет произведение аргументов;
- `SIGN(X)` – возвращает знак аргумента;
- `SUM()` – вычисляет сумму аргументов.

Функции с пустыми скобками могут принимать любое число аргументов. В них можно указывать ссылки на ячейки *перечислением* (A1, A2, B6), *диапазоном* (A1:A10), *блоком* (A1:C5).

Существуют *ключевые* слова, которые могут быть использованы вместо диапазона ячеек:

- ABOVE – диапазон ячеек от активной до начала столбца;
- BELOW – от активной ячейки до конца столбца;
- LEFT – от активной ячейки до начала строки;
- RIGHT – от активной ячейки до конца строки.

В формулах допустимо использование таких знаков операций, как «+» - сложение, «-» - вычитание, «*» - умножение, «/» - деление, «%» - процент, «^» - возведение в степень, и операторов сравнения «=» - равно, «<>» - не равно, «>» - больше, «<» - меньше, «>=» - больше либо равно, «<=» - меньше либо равно.

Для выполнения вставки формулы (функции) в ячейку необходимо установить в нее курсор, из меню *Таблица* выбрать команду *Формула*, в открывшемся диалоговом окне для выполнения простых вычислений в поле *Формула* ввести формулу, а при необходимости в окне *Формат_числа* выбрать требуемый формат. Для более сложных вычислений в поле *Вставить функцию* выбрать нужную функцию.

Например, =SUM(ABOVE) позволит просуммировать числовые значения всех ячеек, начиная от активной до начала столбца. Текст, пустые ячейки и графические объекты будут игнорироваться.

Для просмотра формул в ячейках нужно нажать <Shift>+<F9>, повторное нажатие этой комбинации позволит вернуться в нормальный режим.

Данные таблицы можно отсортировать. Для этого необходимо выделить данные, подлежащие сортировке, вызвать команду *Сортировка* из меню *Таблица*. В открывшемся окне выбрать объект сортировки (столбец1, столбец2 и т.д.), указать критерий сортировки (текст, число, дата) и способ упорядочивания (по возрастанию или убыванию). Для разных столбцов допустимо указание различных критериев сортировки и способов упорядочивания. При этом необходимо учитывать, что каждому значению одной ячейки строки таблицы соответствует значение другой ячейки той же строки таблицы, поэтому при сортировке таких значений следует указывать не один столбец, а несколько. В противном случае может возникнуть несоответствие отсортированных данных исходным. Сортировать можно не только данные таблицы, но и абзацы текста.

После создания таблицы ее можно скопировать в качестве автотекста. Для этого следует выделить таблицу, из меню *Правка* выбрать команду *Копировать*, из этого же меню выбрать команду *Автотекст*, в появившемся диалоговом окне в поле *Имя* ввести имя таблицы и щелкнуть по кнопке *Добавить*. Чтобы вставить таблицу в текст, необходимо установить курсор в место вставки, выбрать из меню *Правка* команду *Автотекст*, в поле *Имя* ввести имя таблицы и нажать кнопку *Вставить*.

Использование стандартных стилей оформления позволяет формировать оглавление документа. Если заголовки документа были оформлены с использованием встроенных стилей Word, то можно составить оглавление, используя команду *Оглавление и указатели* из меню *Вставка*. В появившемся диалоговом окне на вкладке *Оглавление* задать вид оглавления, количество уровней заго-

ложка и т.п. Оглавление будет сформировано после нажатия кнопки ОК в месте, где находился курсор ввода.

Для добавления верхних и нижних колонтитулов, а также корректировки номеров страниц используется команда *Колонтитулы* из меню Вид. После этой выполнения команды текст документа становится временно недоступным для внесения изменений, а в области верхнего и нижнего колонтитулов можно добавлять любую информацию. Повторный выбор команды вернет пользователя в прежний режим работы.

5.5. Вопросы для самоконтроля

1. Дайте определение *«текстовый процессор»*.
2. Приведите классификацию текстовых процессоров.
3. К какой группе текстовых процессоров относится MS Word? Назовите его основные возможности.
4. Каким образом можно вызвать команду в MS Word?
5. Сформулируйте принципы создания и редактирования документа.
6. Охарактеризуйте команды меню *Файл*.
7. Охарактеризуйте команды меню *Правка*.
8. Охарактеризуйте команды меню *Вставка*.
9. Охарактеризуйте команды меню *Формат*.
10. Каким образом можно установить автоматическую расстановку переносов в документе?
11. Каким образом можно задать отступы в документе?
12. Каким образом задать параметры страницы, размер листа и его ориентацию?
13. Какие действия можно выполнять, используя команду *Абзац* меню Формат?
14. Что является основной структурной единицей текстового документа?
15. Какие структурные элементы страницы различают?
16. Каким образом можно установить выравнивание текста в документе?
17. Чем отличается вырезание от копирования, и как выполняются эти действия?
18. Перечислите этапы работы с таблицами.
19. Какие функции используются при организации вычислений в таблицах и почему в некоторых используются пустые скобки?
20. Какие ключевые слова могут быть использованы вместо диапазонов ячеек? Охарактеризуйте их и приведите примеры.

ГЛАВА 6. ОСНОВНЫЕ ПОНЯТИЯ ЯЗЫКА ПАСКАЛЬ

На протяжении уже многих лет среди программистов достаточно популярен язык программирования Паскаль (Pascal). Этот язык был разработан Никлаусом Виртом первоначально для целей обучения программированию вообще, но, благодаря ряду особенностей, получил достаточно широкое распространение и в практической работе. 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.

Как в любом языке программирования, в Паскале присутствуют три составляющие : алфавит, синтаксис и семантика [3].

Алфавит – это фиксированный для данного языка набор основных символов, из которых может состоять текст на этом языке. Никакие другие символы в тексте не допускаются.

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

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

6.1. Алфавит языка Паскаль

Программа на Паскале формируется с помощью конечного набора знаков, образующих алфавит языка. Алфавит Паскаля состоит из 26 прописных и строчных букв латинского алфавита (A – Z и a – z), знака подчеркивания, десятичных (от 0 до 9) цифр и шестнадцатеричных чисел, специальных символов. Шестнадцатеричные числа строятся из десятичных цифр и букв от A до F (или от a до f). При написании программ могут использоваться специальные символы.

Таблица 6.1. Специальные символы языка Паскаль

+ 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*, представленные в таблице 6.2.

Таблица 6.2. Составные символы языка Паскаль

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

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 фигурные скобки { } или их альтернатива – (* *).

6.2. Лексемы языка

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

Слова. Их можно подразделить на *зарезервированные* слова, *стандартные идентификаторы и идентификаторы пользователя.*

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. (`Min`, `Ln`).

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

Паскаль не различает идентификаторы, написанные прописными и строчными символами. Например, `SUM2`, `BL_6` – правильные идентификаторы; `1KL` - ошибка, идентификатор должен начинаться с цифры; `NOMER.DOMA` – ошибка, идентификатор содержит точку.

Константы – это элементы данных, значения которых известны заранее и в процессе выполнения программы не изменяются. Различают такие виды констант, как 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, например, 'A', '45', '?' и т.п.;
- строковые константы, например, 'FFF', 'DOM', 'BOX' и т.п.

Именованная константа – это идентификатор, которому должно быть присвоено значение. Такие константы должны быть описаны в разделе описания констант программы. В Паскале для описания констант используется зарезервированное слово `Const`.

Формат

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

Пример

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

В таблице 6.3 представлен ряд констант, к значениям которых можно обращаться без предварительного определения.

Таблица 6.3. 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

Типизированная константа – это константа, для которой указывается не только значение, но и тип. Типизированной константе можно присваивать только те значения, которые определяются данным типом. *Например*,

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

где `INTEGER` – целочисленный тип.

Переменная – это именованный объект, который может изменять свое значение в процессе выполнения программы.

Каждая переменная характеризуется *именем*, которое является идентификатором, и *типом*.

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;
```

6.3. 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 Паскаль. К ним относятся целочисленные, вещественные, литерные, булевские типы данных и указатели.

Целочисленные типы данных представляют собой значения, которые могут использоваться в математических выражениях. Данные целочисленных типов могут быть представлены как в десятичной, так и в шестнадцатеричной системе, для записи которой используется знак $. Характеристика целочисленных типов данных представлена в таблице 6.4.

Таблица 6.4. 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

Очевидно, что использование переменной типа `Byte` по сравнению с типом `Integer` экономит память 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`.

Пример

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

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 6.5.

Таблица 6.5. 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

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;
```

Булевский тип данных описывается 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. Значение константы или переменной этого типа заключается в апострофы. *Например*, `Var Bukva, Znak, Simvol: char;`

Переменные символьного типа занимают в памяти 1 байт. Использование данных типа `char` в арифметических выражениях запрещено. К литерным зна-

чениям 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 Season =(Spring, Summer, Autumn, Winter);
Var S1, S2: Season;
    Autumn: (September, October, December);
```

В данном примере приведен явно описанный тип данных пользователя Season. Определены их значения – обозначения времен года. Переменные S1 и S2 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. Autumn является переменной этого типа и может принимать значения September, October, December. 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 стандартных 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 Паскаль 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;

файл – последовательность компонентов одного типа и одной длины.

Еще двум структурированным типам – процедурному и типу object (объектному) трудно поставить в соответствие данные в обычном представлении. Авторы не ставят своей целью рассмотреть все возможности языка программирования Паскаль, поэтому ограничиваются рассмотрением в главах 11-12 таких типов данных, как массивы, строки и записи.

6.4. Арифметические выражения и правила их записи

Арифметическое выражение представляет собой набор операндов, соединенных знаками арифметических операций, задающими действия над элементами данных и порядок их выполнения, и возвращает целое или действительное значение. Операндами могут являться константы, переменные и обращения к функциям.

В Паскале могут использоваться такие знаки арифметических операций, как «*» - умножение, «/» - деление, «+» - сложение и «-» - вычитание. Также в арифметических выражениях могут быть использованы функции целочисленного деления DIV и MOD. Операции с использованием этих функций допустимы только для аргументов целочисленного типа (`Integer`, `Byte` и т.п.).

Функция `DIV` возвращает целую часть от деления числа нацело, а функция `MOD` – остаток от деления. Например, `56 DIV 5 = 11` – выделяет целую часть полученного частного; `56 MOD 5 = 1` - выделяет остаток от деления нацело. В арифметических выражениях могут использоваться стандартные функции, представленные в таблице 6.6.

Таблица 6.6. Стандартные функции языка Паскаль

Математическая запись	Запись на 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)`	$-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$
$\|x\|$	`Abs(X)`	

Аргументами этих функций могут быть данные любых числовых типов. Кроме них, при решении задач можно использовать следующие функции [3]:

- получение целой части: `Trunc(X)` – результат имеет целочисленный тип; `Int(X)` – результат имеет вещественный тип;
- получение дробной части: `Frac(X)` – результат вещественного типа;
- округление до целого: `Round(X)` – результат имеет вещественный тип;
- `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.
- 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`.

Для определения четности/нечетности числа используется функция `ODD`. Ее аргументом может быть переменная только целого типа.

`ODD(X)` – возвращает значение булевского типа, равное `True`, если X – нечетное и `False`, если X – четное. Например, `ODD(3)` – возвращает значение `True`, так как аргумент нечетный, `ODD(4)` – вернет значение `False`, `ODD(2+7-1)` – также вернет значение `False`.

Если в выражениях имеются нестандартные функции, они должны быть выражены через стандартные, используя правила математики.

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

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

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

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

Пример

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

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

6.5. Вопросы для самоконтроля

1. Какие символы составляют алфавит языка Паскаль?
2. Как правильно записать идентификатор на языке Паскаль?
3. Дайте определение понятия «лексема» языка Паскаль? Охарактеризуйте виды лексем.
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. Приведите формат описания литерного (символьного) типа данных. Охарактеризуйте этот тип данных.
16. Приведите формат описания перечисляемого типа данных. Охарактеризуйте этот тип данных.
17. Приведите формат описания интервального типа данных. Охарактеризуйте этот тип данных.
18. Поясните на примере различие в описании и использовании перечисляемого и интервального типов данных.
19. Перечислите и охарактеризуйте структурированные типы данных языка Паскаль.
20. Перечислите стандартные функции языка Паскаль.

ГЛАВА 7. СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ

7.1. Характеристика разделов программы

Любая программа реализует алгоритм решения задачи. Она объединяет последовательность действий, выполняемых над определенными данными с помощью определенных операций для реализации определенной цели. Основными характеристиками программы является 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. Имя программы является идентификатором и формируется по соответствующим правилам.

Формат заголовка программы

```
Program <имя программы>;
```

Например,

```
Program Lin;
```

После заголовка следует *программный блок*, состоящий в общем случае из *разделов описаний* подключаемых библиотечных модулей, меток, констант, типов данных, переменных, процедур и функций, и *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.

Охарактеризуем разделы программного блока.

Раздел описания подключаемых библиотечных модулей

Характерной чертой языка Pascal является наличие большого количества стандартных процедур и функций (общее название – подпрограммы). Все они находятся в библиотечных модулях. *Библиотечный модуль* – это результат компиляции одной или нескольких процедур и функций. Модуль имеет имя, упомянув которое в разделе `Uses`, можно получить доступ к каждой из нахо-

дящихся в нем процедур или функций. Раздел описания подключаемых библиотечных модулей всегда расположен после заголовка программы.

Формат

```
Uses<имя модуля1>,<имя модуля2>, …;
```

Пример

```
Uses Crt, Printer;
```

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

Рассмотрим некоторые процедуры модуля `CRT`, которые 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 из диапазона от 1 до 80;
- N2 – номер строки на экране дисплея, целое из диапазона от 1 до 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)`.

Раздел описания меток

Перед любым оператором языка Pascal можно поставить метку, что позволит выполнить обращение к этому оператору с помощью оператора перехода `goto` из любого места программы. Метка состоит из имени и следующего за ним двоеточия. Именем может служить идентификатор или цифра.

Используемые метки должны быть описаны в разделе `Label`.

Формат

```
Label <имя, . . . >;
```

Пример

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

Далее в тексте программы выполняется обращение к меткам с использованием оператора `Goto`.

Пример

```
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. Синтаксис языка Pascal не требует обязательного определения идентификатора типа, так как значения типа 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;
```

Раздел описания процедур и функций

Стандартные процедуры и функции в программе не описываются. В разделе описания процедур и функций должны быть приведены тексты только тех процедур и функций, которые разработаны самим программистом и используются в данной программе. Подробнее процедуры и функции пользователя будут рассмотрены в главе 10.

Раздел операторов

Раздел операторов является основным, так как именно в нем описывается алгоритм решения задачи. Этот раздел начинается со слова `Begin` и заканчивается словом `End` с точкой (`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 («;»).

Пример

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

```
   < оператор;>
End.
```

7.2. Операторы языка Паскаль

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, стоящее в его правой части, и присвоить результат идентификатору переменной, стоящей 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 блок, в котором она описана. Попытка выйти за пределы блока вызывает программное прерывание.

Обычно оператор GOTO применяется для преждевременного выхода из цикла или при отладке программы.

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.
```

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.
```

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 используются операторы Read и Readln.

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.

Операторы записи Write и Writeln производят 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.

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 вывода 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.

7.3. Пример реализации линейного вычислительного процесса

Составить программу для расчета эффективности усовершенствования потока по изготовлению женских блузок.

Экономия времени, %, 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.

Данные для отладки: $T_{ст} = 5890$ с., $T_{нов} = 5574$ с.

Решение задачи представить в виде блок-схемы, текста программы с комментариями и результата ее выполнения.

Решение

Алгоритм решения этой задачи представляет собой реализацию линейного вычислительного процесса и включает выполнение следующих действий:

1. Ввод исходных данных.
2. Расчет снижения затрат времени.
3. Расчет роста производительности труда.
4. Вывод результатов на экран.

Блок схема алгоритма решения задачи представлена на рис. 7.1.

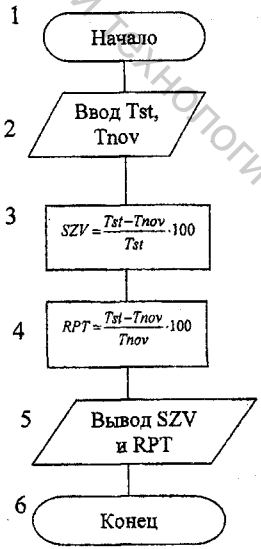


Рис. 7.1. Блок-схема алгоритма решения задачи

Текст программы на Паскале.

```
Program Primer2; {Заголовок программы}
Uses Crt; {Подключение библиотечного модуля CRT}
Var Tst,Tnov:Integer; {Описание переменных целого типа}
    SZV,RPT:Real; {Описание переменных вещественного типа}
Begin {Начало раздела операторов программы}
Clrscr; {Очистка экрана}
{Вывод на экран обращения к пользователю}
   Writeln('Введите старые и новые затраты времени');
   Readln(Tst, Tnov); {Ввод значений Tст и Tнов}
      SZV:=(Tst-Tnov)/Tst*100; {Расчет снижения затрат времени}
      RPT:=(Tst-Tnov)/Tnov*100; {Расчет роста производительности
труда}
   Writeln('Снижение затрат времени = ',SZV:4:1); {Вывод значе-
ния снижения затрат времени с точностью до 1 знака после запятой}
   Writeln('Рост производительности труда = ',RPT:4:1); {Вывод
значения роста производительности труда с точностью до 1 знака после запятой}
Delay(5000) {Задержка на экрана результата выполнения программы на 5 с.}
End. {Конец раздела операторов программы}
```

Результат запуска программы на выполнение.

Введите старые и новые затраты времени
5890
5574
Снижение затрат времени = 5.4
Рост производительности труда = 5.7

7.4. Задания для самостоятельной работы

Вариант 1. Составить линейную программу на языке Паскаль, позволяющую определить размеры суровой ткани (длину и ширину), если известны следующие параметры готовой ткани:

$B_г$ – ширина готовой ткани – 140 см;
$U_у$ – усадка по ширине в отделке – 5%;
$L_г$ – длина куска готовой ткани – 70,2 м;
$U_о$ – усадка по длине в отделке – 3%.

Ширина суровой ткани, см	Длина суровой ткани, м
$B_c = \frac{B_г \cdot 100}{100 - U_у}$	$L_c = \frac{L_г \cdot 100}{100 - U_о}$

Вариант 2. Составить линейную программу на языке Паскаль, позволяющую определить плотность суровой ткани по основе P_{oc} и утку P_{yc}, если известны следующие параметры готовой ткани:

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

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

U_y – усадка по утку в отделке – 5%;

U_o – усадка по основе в отделке – 3%;

Плотность по основе, нит/см	Плотность по утку, нит/см
$P_{oc} = P_{or}\left(1-\frac{U_o}{100}\right)$	$P_{yc} = P_{yr}\left(1-\frac{U_y}{100}\right)$

Вариант 3. Составить линейную программу на языке Паскаль, позволяющую определить длину готового куска ткани $L_г$ с учетом уработки основы длиной $L_{ок}$, идущей на выработку одного куска ткани.

Длина готовой ткани, м	Длина суровой ткани, м	Уработка нитей основы, %
$L_г = L_c\left(1-\frac{U_o}{100}\right)$	$L_c = L_{ок}\left(1-\frac{a_o}{100}\right)$	$a_o = \frac{L_o - L_{oc}}{L_o}100$

где U_o – усадка по длине в отделке – 3%;

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

a_o – уработка нитей основы, %.

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

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

Вариант 4. Составить линейную программу на языке Паскаль, позволяющую определить ширину заправки ткани по берду $B_з$, если ширина суровой ткани B_c= 145 см.

Ширина заправки ткани по берду	Уработка нитей по утку, %
$B_з = \frac{B_c}{\left(1-\frac{a_y}{100}\right)}$	$a_y = \frac{L_y - L_{yc}}{L_y}100$

L_y – длина распрямленной нити утка из образца ткани, L_y=10,8 см;

L_{yc} – длина образца ткани вдоль утка, $L_{yc} = 10$ см.

Вариант 5. Составить линейную программу на языке Паскаль, позволяющую определить длину обычной и ажурной основы $L_{ок}$, необходимой для выработки кусков ткани заданной длины L_c.

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

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

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

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

Вариант 6. 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 7. Составить программу на языке Паскаль для определения расчетного и фактического количества рабочих 3 разряда, необходимого для выполнения технологически неделимой операции, если такт потока τ составляет 75 с, затрата времени на операцию T = 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 $N_p = \frac{T}{\tau}$.

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 8. 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 тарифных 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 9. Составить 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 $К_{эг} = К_р * (К - Л) + К_{эн} * К$,

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 $К_{эс} = К_{эг} / 365$.

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

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

Вариант 10. Составить 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 t 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
$M = \frac{R}{\tau}$	$N_р = \frac{T}{\tau}$	округлить $N_р$ до целого	$K_{исп} = \frac{N_р}{N_ф}$	$П_т = \frac{M}{N_ф}$

ГЛАВА 8. СТРУКТУРНЫЕ ОПЕРАТОРЫ ЯЗЫКА ПАСКАЛЬ. СОСТАВНОЙ И УСЛОВНЫЙ ОПЕРАТОРЫ

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*.

8.1. Составной оператор

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.

8.2. 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 [6].

В разветвляющихся вычислительных процессах отдельные этапы вычислений (операторы) выполняются не всегда в одном и том же порядке. В зависимости от условий, проверяемых по ходу вычислений, выбираются для исполне-

ния различные их последовательности. Поэтому в программе должно содержаться указание о том, в каком случае надо выбирать для исполнения тот или иной оператор. Это указание формулируется в виде отношения, например, условия $x > y$. Если это условие истинно, то выполняется соответствующий этому условию оператор, в противном случае, если условие ложно, 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 за If.

Условие может быть простым и сложным. В последнем случае используются логические операции AND, OR, NOT, например, If (A>3) And (A<5).

Блок-схемы полной и неполной условной конструкции подробно рассмотрены в главе 5 на рис. 5.3.

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. заключаются 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. Допускается до 7 вложений оператора IF друг в друга.

Рассмотрим использование операторов условия для программирования разветвляющихся вычислительных процессов.

Пример 8.1

Составить блок-схему и программу на языке Паскаль для вычисления значения функции Y в зависимости от x по формуле

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

Решение

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 рисунке 8.1.

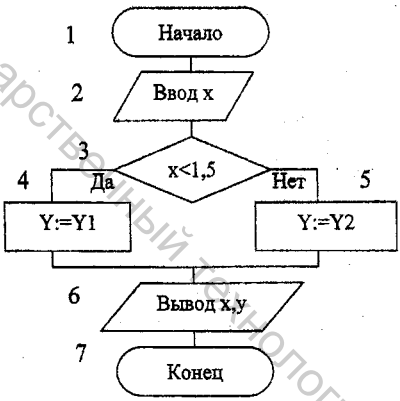


Рис.8.1. Блок-схема вычисления значения функции Y

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 8.2

Составить программу на языке Паскаль для определения степени укорочения волокон в процессе чесания. Укорочение волокон в процессе чесания, %, вычисляется 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 приведены ниже.

η	Оценка
$\leq 15\%$	Отлично
$15 < \eta \leq 18\%$	Хорошо

η	Оценка
$18 < \eta \leq 21\%$	Удовлетворительно
$> 21\%$	Неудовлетворительно

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

Решение задачи представить в виде блок-схемы, текста программы с комментариями и результата ее выполнения.

Решение

Решение данной задачи представляет собой реализацию алгоритма разветвляющейся структуры со сложным логическим условием.

Алгоритм:

1. Ввод длины волокна до и после чесания l_1 и l_2.
2. Расчет степени укорочения волокон η.
3. Проверка: если $\eta \leq 15\%$, то степень укорочения отличная, если $15 < \eta \leq 18\%$ - хорошая, $18 < \eta \leq 21\%$ - удовлетворительная, $\eta > 21\%$ - неудовлетворительная.
4. Вывод на экран значения степени укорочения волокон.

Блок-схема решения задачи представлена на рис. 8.2.

Текст программы на Паскале.

```
Program Primer3; {Заголовок программы}
Uses Crt; {Раздел описания используемых библиотечных модулей}
Var L1, L2, Nu:Real; {Описание переменных}
Begin {Начало раздела операторов программы}
Clrscr; {Очистка экрана}
     Writeln('Введите длину волокна до и после чесания');
       Readln(L1,L2); {Ввод исходных данных}
     Nu:=(L1-L2)/L1*100; {Вычисление степени укорочения}
     If Nu<=15 Then Writeln('Отлично') Else
       If Nu<=18 Then Writeln('Хорошо') Else
         If Nu<=21 Then Writeln('Удовлетворительно')
              Else Writeln('Неудовлетворительно');
     Writeln('Степень укорочения волокон Nu - ',Nu:4:2);
Delay(7000) {задержка результата на экране на 7 с.}
End. {конец раздела операторов}
```

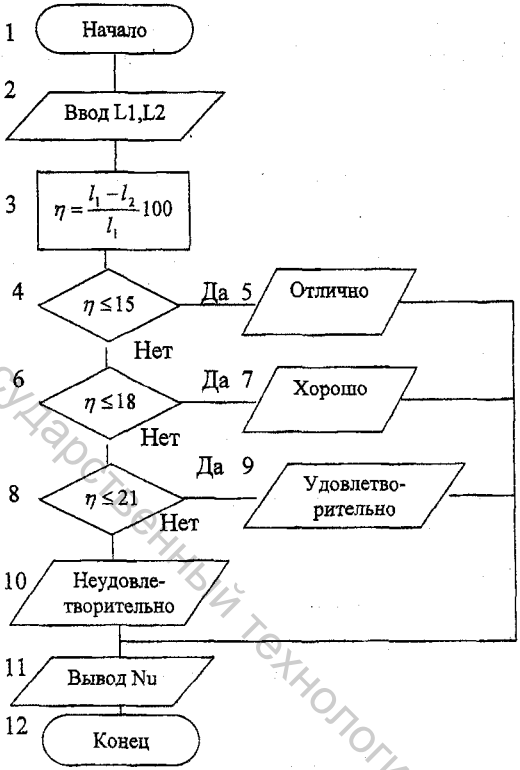


Рис. 8.2. Блок-схема алгоритма решения примера 8.2

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 [6]. Блок-схема оператора Case представлена на рисунке 8.3.

Формат

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

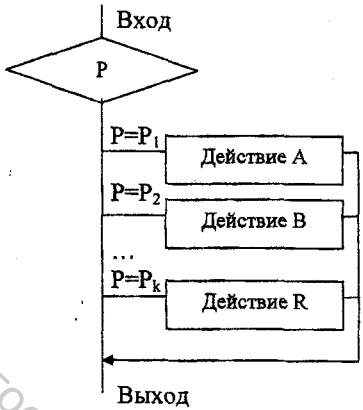


Рис. 8.3. Блок-схема оператора Case

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	Селектор перечисляемого типа

`Case I of`
`  1:Z:=A+B;`
`  2:Z:=A+10;`
`  3:Z:=A+100;`
`end;`
	`Case Season of`
	`  Winter: Writeln('WINTER');`
	`  Spring: Writeln('SPRING');`
	`  Summer: Writeln('SUMMER');`
	`  Autumn: Writeln('AUTUMN')`
	`End;`

Пример 8.3

Составить программу для определения размера премии работникам предприятия, если установлен следующий размер премирования в зависимости от стажа работы: 0-1 год – 0,5 МЗП; 2-5 лет – 1 МЗП; 6-10 лет – 2 МЗП; 11-15 лет – 3 МЗП; 16-20 лет – 4 МЗП; Свыше 20 лет – 5 МЗП. Найти размер премии, полученной работником, отработавшим: а) 7 лет; б) 12 лет; в) 25 лет. Минимальная заработная плата – 65000 рублей.

Решение

```
Program Premia;
Uses Crt;
Var X,Mzp:Integer;
          P:Real;
Begin
```

```
Clrscr;
Writeln('Введите стаж работы');
Readln(X);
Writeln('Введите МЗП');
Readln(Mzp);
Case X Of
0..1:P:=0.5*Mzp;
2..5:P:=Mzp;
6..10:P:=2*Mzp;
11..15:P:=3*Mzp;
16..20:P:=4*Mzp
Else P:=5*Mzp
End;
Writeln('Премия=',P:4:1);
Delay(3000);
End.
```

8.3. 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, а 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 в случае выполнения условий	Повышение производительности труда в случае невыполнения условий
$\eta = \frac{2Kc_2 - Kc \cdot Kc_2 - Kc}{2Kc_2 - Kc \cdot Kc_2 + Kc} \cdot 100$	$\eta = \frac{Kc_1 + Kc_2 - Kc_1 \cdot Kc_2 - Kc}{(1-Kc)(Kc_1 + Kc_2 - Kc_1 \cdot Kc_2) + Kc} \cdot 100$

Определить, повысится ли производительность труда, если Кс=0,08, $Кс_1$=0,055, $Кс_2$=0,05. Найти значение η и вывести его на экран.

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	$Ч_{пл}$ – условная численность в плановом периоде	Доля прироста объема производства за счет повышения производительности труда
$\Delta П_т = \frac{У_ч}{Ч_{пл} - У_ч} \cdot 100$	$Ч_{пл} = Ч_р\left(1 + \frac{P_п}{100}\right)$	$\Delta M = \frac{\Delta П_т}{P_п} \cdot 100$

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

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

Неровнота, вносимая в пряжу прядильной машиной, %	Идеальная неровнота с учетом условий прядения, %
$CV_м = \sqrt{CV_ф^2 - CV_{ид}^2}$	$CV_{ид}^2 = CV_р^2 + 10{,}44\frac{(E-1)d^2}{T_р}$

где $CV_ф$ – фактическая неровнота пряжи по Устеру, %;
$CV_{ид}$ – идеальная неровнота с учетом условий прядения, %.
$CV_р$ – неровнота ровницы по Устеру, %;
E – вытяжка в прядении;
T – линейная плотность ровницы, текс;
d – диаметр волокна, мкм.

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 $P_{у/х}$=6,29, $P_{х/у}$=0,113.

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

$$r = \pm\sqrt{P_{у/х} \cdot P_{х/у}},$$

где $P_{у/х}$ и $P_{х/у}$ – 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.

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 В – 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 В=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 рассчитать 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. На складе швейного предприятия имеются зимние женские пальто, изготовленные в различное время. Варианты представлены ниже.

Вариант 1	Вариант 2	Вариант 3
Количество – 10 шт. Год выпуска – 2001 Цена – 135000 руб.	Количество – 5 шт. Год выпуска – 2004 Цена – 186000 руб.	Количество – 10 шт. Год выпуска – 2007 Цена – 320000 руб.

Процент уценки зависит от года выпуска: если продукция выпущена до 2003 года включительно, то уценка равна 50%, если до 2005 – то 10%, если после 2005 года – то уценки нет. Определить стоимость товара по каждому варианту с учетом уценки.

уценка = процент уценки * цена	стоимость = количество * цена – уценка

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, %

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

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.

Начислено = Коэффициент начислений * минимальный размер начислений.

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

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

Тарифная заработная плата = Минимальная базовая величина * Тарифный коэффициент.

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

Минимальная базовая величина = 62000 рублей.

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 представлены ниже:

Разряд	2	3	4
Стоимость 1 секунды, 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 операцию = 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 9. СТРУКТУРНЫЕ ОПЕРАТОРЫ ЯЗЫКА ПАСКАЛЬ. ОПЕРАТОРЫ ПОВТОРА

Операторы повтора представляют третью группу структурных операторов и используются при организации циклов.

Цикл – это многократное использование одного и того же вычисления с изменяющимися исходными параметрами. Конкретные программные реализации циклов называются циклическими программами.

Различают два типа циклов:

- *цикл со счетчиком*, когда число повторений цикла известно заранее;
- *итерационный цикл*, 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 оператора присваивания, есть сумма элементов конечного ряда, U – значение текущего элемента конкретного ряда, S, стоящее слева от оператора присваивания, - последующее (определяемое) значение суммы конечного ряда.

В случае вычисления произведения формула имеет вид: P:= P*U, где P, стоящее справа от оператора присваивания, есть произведение элементов конечного ряда, U – значение текущего элемента конкретного ряда, P, стоящее слева от оператора присваивания - 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 или произведение при этом вычисляются традиционно: S:=S+U или P:= P*U.

3. *Табулирование функции* – получение таблицы значений функции на заданном интервале $x\in[x_{min}; x_{max}]$ при дискретном изменении x, равном dx.

Для реализации циклов в Паскале используются три оператора повтора: `For` – для реализации циклов, число повторений которых известно заранее (циклов со счетчиком); `Repeat` и `While` – используются как в итерационных циклах, так и в циклах со счетчиком.

9.1. Оператор повтора For

Оператор повтора `For` состоит из заголовка и тела цикла и 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. Параметр цикла в процессе выполнения циклической части программы принимает следующие значения:

для первого формата: N1, N1+1, N1+2,... N2;

для второго формата: N1, N1-1, N1-2,...N2.

Отсюда следует, что в первом случае (при `For ... to`) значение N1 должно быть меньше или равно N2, во втором (при `For ... downto`) – значение N1 больше или равно N2, т.е. 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 рисунке 9.1.

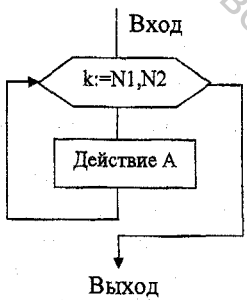


Рис.9.1. Блок-схема оператора For

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 9.1.

Вычислить сумму элементов конечного ряда:

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

Решение

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 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 вычисления сумм элементов конечного ряда сводится к циклу, когда при каждом прохождении цикла номер члена суммы изменяется на 1, а сумма изменяется на ее n-ый член, т.е.

$$S_n = S_{n-1} + U_n \qquad (9.1)$$

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

Формула (9.1) 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 примера 9.1 представлена на рисунке 9.2.

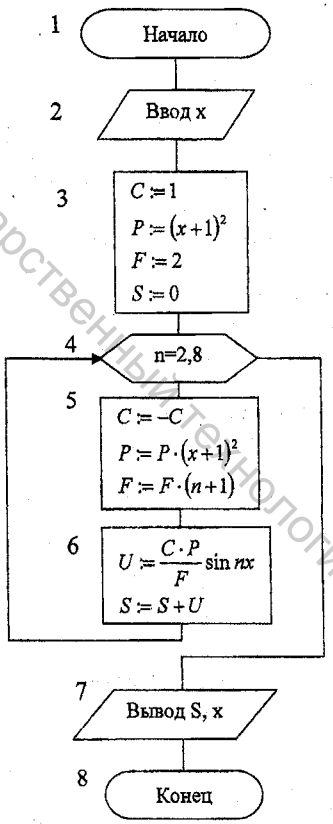


Рис. 9.2. Блок-схема вычисления суммы элементов ряда

Ниже приводится программа вычисления суммы элементов ряда.

```
Program Cikl1;
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=',
N:7:5);
Readln;
End. {конец программы}
```

9.2. 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 постусловием, 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 имеет значение FALSE, тело цикла выполняется еще раз, если результат имеет значение TRUE – происходит выход из цикла. Блок-схема оператора повтора Repeat представлена на рис. 9.3.

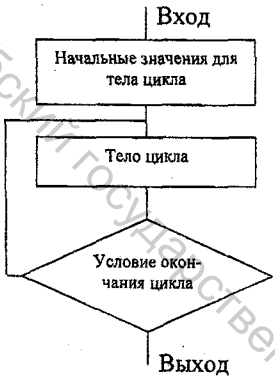


Рис. 9.3. Блок-схема оператора Repeat

Оператор Repeat имеет три характерные особенности:

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

По крайней мере, один из операторов, находящихся в теле цикла, должен влиять на значение условия, иначе цикл будет бесконечным.

Пример

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

Результат: 5, 6, 7, 8

Пример 9.2

Составить с объяснениями блок-схему и программу вычисления суммы элементов ряда с заданной точностью.

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

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.

Решение

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 рисунке 9. 4.

Пояснения к блок-схеме.

Блок 1 – начало.

Блок 2 – ввод значений x, ε.

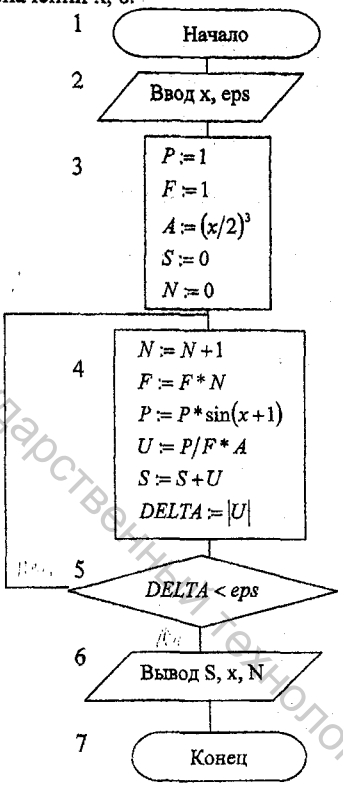


Рис. 9.4. Блок-схема вычисления суммы S

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. Пере-

менная 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.

```
Program Cikl2;
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 окончания цикла}
Until Delta<EPS;
{вывод результата на дисплей}
Writeln('При      x=',x:4:2,'          сумма',      n:2,'
элементов=',S:7:5);
Readln;
End. {конец программы}
```

9.3. Оператор повтора While

Оператор `While` является самым мощным оператором повтора. Любой цикл, построенный с помощью `For` или `Repeat`, может быть построен и с помощью `While`.

Формат

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

Этот оператор аналогичен оператору `Repeat`, но проверка условия выполнения тела цикла производится в самом начале оператора. Здесь выражение представляет собой условие, при котором цикл повторяется. Перед каждым выполнением тела цикла вычисляется значение выражения условия. Если резуль-

тат 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.

Блок-схема оператора While представлена на рис. 9.5.

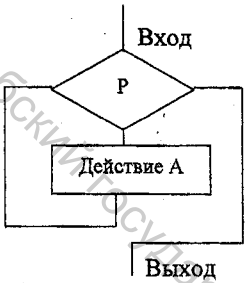


Рис.9.5. Блок-схема оператора While

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

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 9.3.

Разработать схему и программу на языке Паскаль решения задачи вычисления значений функции Y=f(x), заданной на отрезке [a, b], шаг приращения аргумента dx (табулирование функции).

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

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.`

Решение

Алгоритм решения этой задачи включает в себя следующие этапы:

1. Ввод исходных данных, в качестве которых выступают a, b – границы интервала табулирования, dx – приращение аргумента и коэффициент t.
2. Задание начального значения аргумента функции x: = a.
3. 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 выполняется, тело цикла повторяется, в противном случае происходит выход из цикла.

Блок-схема алгоритма решения задачи приведена на рис. 9.6.

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`.

Текст программы решения задачи на языке Паскаль.

```
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}
Writeln('Таблица значений функции');
```

```
x:=a; {начальное значение x=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);
x:=x+dx; {получение нового значения аргумента}
 End; {конец цикла}
Readln;
End.
```

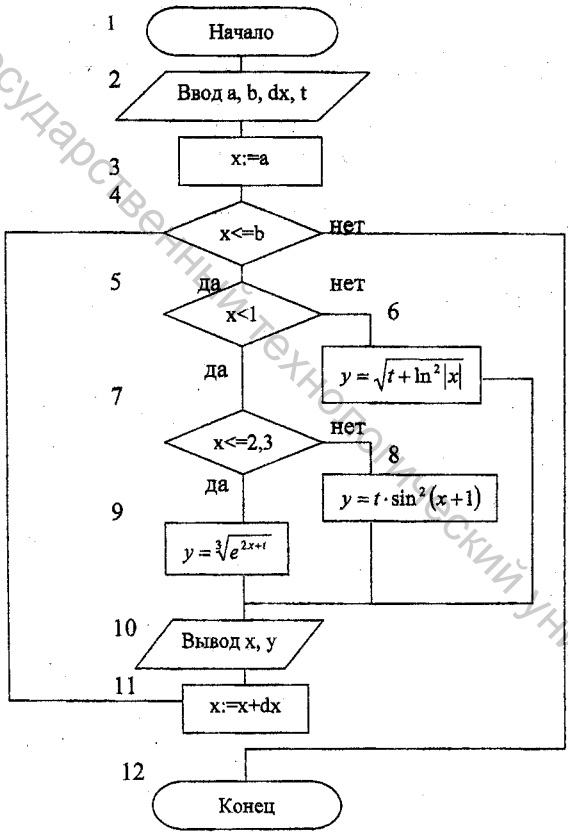


Рис.9.6. Блок-схема табулирования функции Y=f(x)

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;

9.4. Задания для самостоятельной работы

- **Вычисление 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.

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.

Вариант 2. Количество влаги W, необходимое для насыщения воздуха в прядильном зале предприятия, вычисляется по формуле

$$W = 5{,}336 \cdot 1{,}059^x,$$

где x – температура воздуха, °C.

Найти количество влаги, необходимое для насыщения воздуха в прядильном зале предприятия при температуре воздуха от 5 до 40°C, шаг изменения температуры dx=2,5°C.

Вариант 3. Максимальная разрывная нагрузка волокна мериносовой шерсти (по Кронахеру) вычисляется по формуле

$$P = 0{,}0056D^2 + 0{,}503D - 3{,}77,$$

[1] Функциональные зависимости y=f(x), предлагаемые для табулирования, 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..

где D – диаметр волокна, мкм.

Определить максимальную разрывную нагрузку волокна мериносовой шерсти диаметром от 16 до 24 мм. Шаг изменения диаметра волокна шерсти dD=1 мм.

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:

$$y = \begin{cases} 0.28*x^3 + 5.39*x^2 - 22.78*x + 131.9 & x < 2000 \\ 1.34*x + 91.3 & x \geq 2000, \end{cases}$$

где x – год выпуска продукции.

Определить индексы производства непродовольственных товаров за период с 1990 по 2005 год.

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

$$y = \begin{cases} 18.59*x + 60{,}45 & x \leq 1998 \\ 0.084*x^3 - 1.4*x^2 + 5.5*x + 103.2 & x > 1998, \end{cases}$$

где x – год выпуска продукции.

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:

$$y = \begin{cases} 8.69*x + 77.67 & x \leq 1999 \\ 0.0091*x^3 + 0.156*x^2 + 0.956*x + 110.57 & x > 1999, \end{cases}$$

где x – год выпуска продукции.

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:

$$y=\begin{cases}7.01*1.108^{x} & x\le 1999\\ 0.0032*x^{4}-0.086*x^{3}+0.75*x^{2}-1.97*x+11.06 & x>1999,\end{cases}$$

где x – год реализации продукции.

Найти рентабельность реализации промышленной продукции за период с 1992 по 2005 год.

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

$$y=\begin{cases}5.95*x^{3}-110.61*x+1649 & x\le 1999\\ -18.9*x+1343.3 & x>1999,\end{cases}$$

где x – год.

Найти среднегодовую численность производственно-промышленного персонала за период с 1990 по 2005 год.

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

$$y=\begin{cases}74.1*1.06^{x} & x\le 1999\\ 0.0053*x^{2}+1.24*x+93.68 & x>1999\end{cases}$$

где x – год выпуска продукции.

Определить объемы выпуска продукции в сопоставимых ценах за период с 1992 по 2005 год.

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

$$y=\begin{cases}-3.65*x^{3}+35.6*x^{2}-103.75*x+281.01 & x\le 1999\\ -4.91*x+249 & x>1999,\end{cases}$$

где x – год выпуска продукции.

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:

$$y=\begin{cases}-21.8*x+203 & x\le 1998\\ 0.186*x^{3}-3.139*x^{2}+18.67*x+225.73 & x>1998,\end{cases}$$

где x – год выпуска продукции.

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:

$$y=\begin{cases}1.29*x^2-0.714*x+36.6 & x\le 1999\\ 0.075*x^2-4.29*x+74.85 & x>1999\end{cases}$$

где x – год выпуска продукции.

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:

$$y=\begin{cases}1.25*x+3.05 & x\le 1999\\ 0.011*x^3-0.19*x^2+0.69*x+8.19 & x>1999\end{cases}$$

где x – год выпуска продукции.

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 10. ПОДПРОГРАММЫ В ЯЗЫКЕ ПАСКАЛЬ

10.1. Понятие подпрограммы. Виды подпрограмм

В практике программирования часто встречаются ситуации, когда одну и ту же группу операторов, реализующих определенную цель, необходимо 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 Паскаль для организации подпрограмм используются *подпрограммы-процедуры* и *подпрограммы-функции*, которые, в зависимости от целей программирования, могут размещаться следующим образом:

- головная программа и подпрограммы расположены 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 Паскаль разделяются на встроенные и описанные пользователем. *Встроенные* (стандартные) процедуры и функции являются частью языка и могут вызываться по имени без предварительного определения в разделе описаний. Различают 9 основных групп встроенных процедур и функций. Для обозначения типов данных аргументов используются сокращения: IBR – целочисленный (I), байтовый (B), вещественный тип (R); S – любой из скалярных типов, кроме вещественного.

1. Арифметические: `Abs(IBR)`, `Arctan(IBR)`, `Cos(IBR)` и т.д.
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(T)`, `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.

10.2. Процедуры пользователя

Использование подпрограмм пользователя обеспечивает возможность организации работы нескольких программистов над одной программой, проведение отладки отдельных блоков и только потом всей программы в целом, значительно экономит память, т.к. многократно используемый участок заносится в память только один раз, упрощает 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. 10.1:

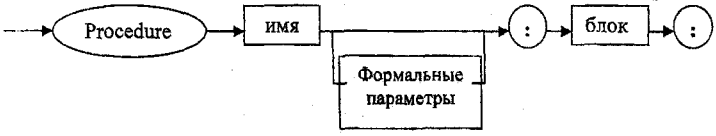


Рис.10.1. Синтаксическая диаграмма подпрограммы-процедуры

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;` .

10.3. Параметры и переменные в подпрограммах

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 заключается в том, что 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 соответственно 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);`

параметр-значение параметр-переменная

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 10.1. 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 их 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 выполнения поиска целесообразно объявить процедурой.

Если программа имеет модульную структуру и состоит из ряда вложенных друг в друга блоков, то идентификаторы, описанные в определенных модулях, называются *локальными* и могут быть использованы только в той части программы, где они описаны. Идентификаторы, описанные в основной программе, являются *глобальными* и могут использоваться во всех вложенных блоках (процедурах и функциях). 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.

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 **10.1** 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.

10.4. Функции пользователя в языке Паскаль

Хотя набор встроенных функций Паскаля достаточно широк, он вряд ли может удовлетворить требованиям каждого программиста. Поэтому пользователю предоставляется 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 присутствовать по крайней мере один оператор, присваивающий идентификатору функции значение. Если таких опера-

торов 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. 10.2:

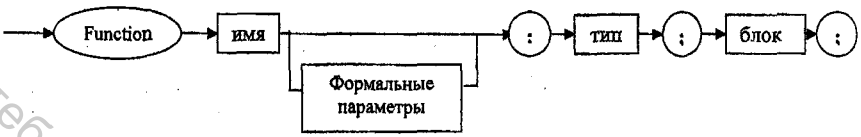


Рис.10.2. Синтаксическая диаграмма подпрограммы-функции

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)`.

Функция без параметров. Обратимся к условию примера 10.1, в котором необходимо вычислить большее 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 10.1, в котором необходимо вычислить большее из двух значений U и V, где `U = max(x+y), V = max(0.5, U)` и составим программу с использованием подпрограммы-функции с параметрами.

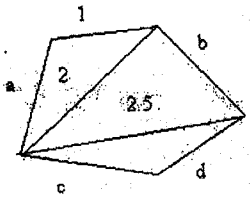
```
Program Max_E;
     Var x,y,u,v:real;
(*************************************)
     Function Max(a,b:real):real;
      Var S:real;
      Begin
       If a>b Then S:=a Else S:= b;
       Max:=S;
      End;
(*************************************)
     Begin
     Writeln(' Введите x,y');
     Readln(x,y);
     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.

10.5. Примеры использования подпрограмм при решении задач

Пример 10.2

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;
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.
```

Использование подпрограммы-функции

```
Program SF;
 Var a,b,c,d,S:real;
(************************)
Function
GERON(x,y,z:real):real;
Var P, SG:real;
Begin
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.
```

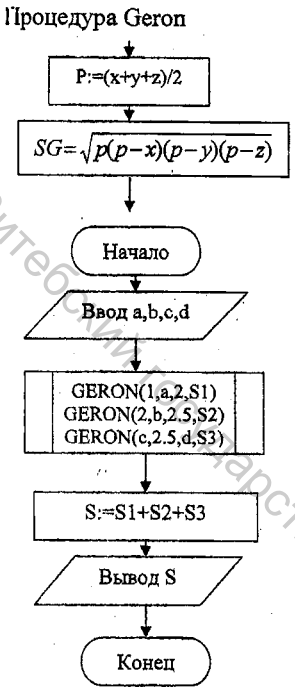


Рис. 10.3. Блок-схема решения задачи с использованием процедуры

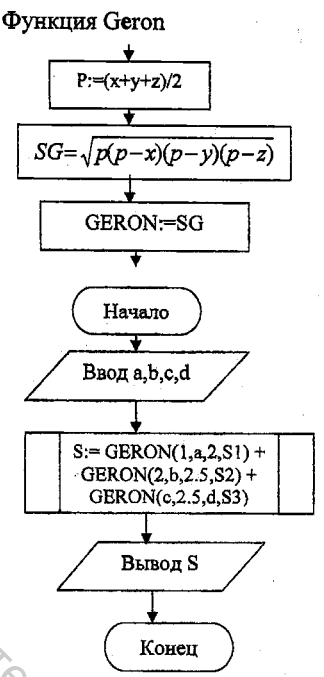


Рис. 10.4. Блок-схема решения задачи с использованием функции

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.

10.6. 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 А содержит процедуру В, 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 решение 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.

10.7. Задания для самостоятельной работы

Составить программы на языке Паскаль 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.

Шифр	Наименование материальных средств	Первоначальная стоимость, $C_{пер}$, руб.	Сумма износа, $A_{мес}$, %	Норма амортизации, H_a, %
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

Вычисление износа за месяц и остаточной стоимости оформить через подпрограмму.

Износ за месяц, руб.	Остаточная стоимость, руб.
$A_{мес} = \frac{C_{пер} * H_a}{12*100}$	$O_{ост} = C_{пер} - C_{изн} - A_{мес}$

Вариант 4. Определить выпуск в смену по моделям, если в потоке обрабатываются 3 модели. 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 (ТЭП) 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, М – 369 единиц. Расчет ТЭП оформить через подпрограмму. Показатели:

Выпуск каждой модели	Такт потока по модели	Расчетное количество рабочих
$M_i = \frac{M}{\Sigma m_i} \cdot m_i$. Полученное значение округлить до целого числа	$t_i = \frac{R_i}{M_i}$	$N_p = \frac{T_i}{t_i}$

Вариант 7. Определить, используя подпрограмму, годовую экономию текущих затрат по каждому виду работ и экспериментальному цеху в общем для каждого варианта механизации. Вычисленные значения вывести на экран.

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 %.

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, %	Рост производительности труда, %
$СЗВ = \frac{T_{ст} - T_{нов}}{T_{ст}} \cdot 100$	$РПТ = \frac{T_{ст} - T_{нов}}{T_{нов}} \cdot 100$

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.

Показатели:

Расчетный такт потока	Мощность потока в смену, ед. (округлить до целого)	Расчетное количество рабочих	Фактическое количество рабочих Nф	Коэффициент использования рабочего времени	Производительность труда 1 рабочего
$\tau_p = \frac{\%откл}{100} \cdot \tau$	$M = \frac{R}{t_{p_i}}$	$N_p = \frac{T}{t_{p_i}}$	Принимается округлением расчетного количества до целого	$K_{исп} = \frac{N_p}{N_ф}$	$П_T = \frac{M}{N_ф}$

Мощность рациональна, если $K_{исп}$=1, производительность труда больше средней, а рекомендуемое количество рабочих составляет 50-90 человек.

ГЛАВА 11. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ. МАССИВЫ

11.1. Понятие массива. Способы описания массивов

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 *структурированных* или *производных* типов в общем случае представляют собой нетривиальную структуру, то есть 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 Паскаль 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. *Массив* – это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип.

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 Паскаль 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, двумерные – 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. 11.1.

Рис. 11.1. 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 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; n: Gran;
     c: ch_M; d: ch_R;
```

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.

11.2. Действия над элементами массивов

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, и два двумерных массива (V и W), содержащих элементы вещественного типа.

```
Var A, B: array[1..5] of integer;
    C, D: array [1..10] of real;
    V, W: array [1..4,1..4] 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 или V[1,1]:=0; V[1,2]:=0; и т.д. 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;
```

```
For I:=1 To 4 Do
For J:=1 To 4 Do
V[I,J]:=0;
```

Ввод-вывод элементов массива. Паскаль 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,' элемент
массива');
Read(A[I]);
End;
```

```
For I:=1 To 4 Do
Begin
Writeln('Введите ',I,' строку');
For J:=1 To 4 Do
Read(V[I,J]);
End;
```

Вывод элементов массива производится аналогично, но с использованием оператора Write (Writeln).

```
Writeln('Вывод массива A');
For I:=1 To 5 Do
Write(A[I]:4:2,' ':2);
Writeln;
```

```
Writeln('Вывод массива V');
For I:=1 To 4 Do
Begin
 For J:=1 To 4 Do
   Write(V[I,J]:4:2,' ':2);
 Writeln;
End;
```

Копирование элементов из одного массива в другой. Для выполнения этого действия необходимо, чтобы массивы, участвующие в этой операции, были описаны одинаково. Например, B:=A; - верно, так как B и A имеют одинаковый тип компонент; A:=C; - неверно, т.к. C имеет компоненты типа real, а A – типа integer и массивы A и C разной длины). Или, используя цикл:

```
For I:=1 To 5 Do Begin
  B[I]:=A[I];
  C[I]:=B[I] End;
```

```
For I:=1 To 4 Do
For J:=1 To 4 Do
W[I,J]:=V[I,J];
```

Поиск в массиве элементов, удовлетворяющих некоторым условиям. Основные методы поиска будут подробно рассмотрены в главе 12. Здесь рассмотрим наиболее часто выполняемые действия. Пусть надо выяснить, сколько элементов в массивах C и W имеют нулевое значение.

```
K:=0;
For I:=1 To 10 Do
If C[I]=0 Then K:=K+1;
```

```
K:=0;
For I:=1 To 4 Do
For J:=1 To 4 Do
If W[I,J]=0 Then K:=K+1;
```

После выполнения цикла переменная K будет содержать количество элементов с нулевым значением.

Поиск минимального (максимального) элемента массива рассмотрим на примере поиска минимального элемента в одномерном массиве C и максимального – в двумерном массиве W.

```
Min:=C[1];
For I:=1 To 10 Do
If C[I]<Min Then Min:=C[I];
```

```
Max:=W[1,1];
For I:=1 To 4 Do
For J:=1 To 4 Do
If W[I,J]>Max Then Max:=W[I,J];
```

Вычисление суммы (произведения) элементов массива.

Найти сумму (S) и произведение (P) элементов массивов C и W.

```
S:=0; P:=1;
For I:=1 To 10 Do
Begin
S:=S+C[I];
P:=P*C[I];
```

```
S:=0; P:=1;
For I:=1 To N Do
For J:=1 To M Do
Begin
S:=S+W[I,J];
```

```
End;
```

```
P:=P*W[I,J];
End;
```

Перестановка элементов массива осуществляется с использованием вспомогательной переменной того же типа, что и базовый тип массивов.

Поменять местами первый и четвертый элемент массива A

```
Per:=A[1];
A[1]:=A[4];
A[4]:=Per;
```

Поменять местами первый, первый и четвертый, четвертый элемент массива W

```
Per:=W[1,1];
W[1,1]:=W[4,4];
W[4,4]:=Per;
```

Рассмотрим основные действия над элементами массивов на примерах.

Пример 11.1. Обработка одномерного массива.

Вычислить % дней в месяце, когда температура опускалась ниже 0^0C.

Решение

Для упрощения вычислений введем 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 Паскаль.

```
Program Cold;
Uses CRT;
Const all_d =31;                {описание константы}
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}
cold_d:=0; {подсчет в % количества дней с температурой ниже 0^0}
  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 11.2. Обработка двумерного массива

В таблице представлены сведения о длинах концевых остатков, м, куско[в] ткани артикула 3С170Д41. Найти максимальную и среднюю длины концевы[х] остатков по всем кускам ткани. Значения минимальной и средней длины конце[в]ых остатков, а также исходную матрицу длин концевых остатков вывести н[а] экран.

Фабричный номер куска	Длина концевого остатка, м		
	170-96	176-100	182-100
8801	0,25	2,42	1,3
8724	0,45	2,62	1,5
8736	2,55	0,87	3,95
8610	2,95	1,27	0,15
8782	2,8	1,12	0,15

Решение

Алгоритм решения данной задачи представляет собой обработку матрицы размера 5x3, содержащей элементы вещественного типа.

1. Ввести значения элементов массива
2. Вычисление суммы элементов массива для определения средней длины концевого остатка.
3. Вычисление средней длины концевого остатка.
4. Поиск максимальной длины концевого остатка.
5. Вывод на экран исходной матрицы, максимальной и средней длин концевых остатков.

Текст программы на Паскале.

```
program ostatki;
uses crt;
type matr=array[1..5,1..3] of real;
var A:matr;
    i,j:integer;
    max,sr,s:real;
begin
clrscr;
 writeln('Ввод матрицы значений концевых остатков');
   for i:=1 to 5 do
     begin
       writeln('Введите ',i,' строку=>');
        for j:=1 to 3 do
         read(a[i,j])
     end;
s:=0; {вычисление средней длины концевых остатков}
   for i:=1 to 5 do
     for j:=1 to 3 do
       s:=s+a[i,j];
 sr:=s/(5*3);
max:=a[1,1]; {Поиск максимальной длины концевого остатка}
   for i:=1 to 5 do
     for j:=1 to 3 do
       if a[i,j]>max then max:=a[i,j];
 writeln('Исходная матрица');
   for i:=1 to 5 do
     begin
```

```
      for j:=1 to 3 do
        write(a[i,j]:4:2,' ':2);
      writeln
    end;
writeln('Средняя длина концевого остатка - ',sr:4:2);
writeln('Максимальная длина концевого остатка - ',max:4:2);
readln
end.
```

11.3. Процедуры и функции обработки массивов

Рассмотрим основные операции по обработке данных, организованных в массивы, на уровне подпрограмм-процедур и подпрограмм-функций, описанных для формальных параметров. Для этого опишем следующие массивы: матрицы A и B размера MxN и одномерные массивы C и D длиной N ($M, N \le 10$). Кроме того, опишем в разделе описания переменных основной программы переменные I и J, которые будут являться параметрами циклов. Это позволит не описывать их каждый раз в подпрограммах.

```
Type Matr = Array[1..10,1..10] Of Real;
     Mas = Array[1..10] Of Real;
Var A, B, C: Matr;
    F, D: Mas;
    I, J: Integer;
```

Ввод элементов массива.

Одномерный массив

```
Procedure Vvod(N:Integer; Var
W:Mas);
Begin
For I:=1 To N Do Begin
Write('Введите ',I, 'элемент-');
Read(W[I])
End
End;
```

Обращение: `Vvod(10,D);`

Ввод элементов матрицы по строкам

```
Procedure Vvod(M,N:Integer; Var
V:Matr);
Begin
For I:=1 To M Do Begin
Writeln('Введите ', I,' строку');
  For J:=1 To N Do
    Read(V[I,J])
End
End;
```

Обращение: `Vvod(4,5,A);`

Вывод элементов массива.

```
Procedure Vyvod(N:Integer; Var
V:Mas);
Begin
For I:=1 To N Do
  Write(W[I,J]:4:2,' ':2);
Writeln
End;
```

```
Procedure Vyvod(M, N:Integer; Var
V:Matr);
Begin
For I:=1 To M Do Begin
  For J:=1 To N Do
   Write(V[I,J]:4:2,' ':2);
 Writeln
End
End;
```

Обращение: `Vyvod(10,D);`

Обращение: `Vyvod(5,5,B);`

Вычисление суммы (произведения) элементов массива.

Вычисление произведения или суммы элементов массива производится с использованием подпрограммы-функции, так как результатом работы является одно значение суммы.

```
Function Summa(N:Integer;
V:Mas):Real;
    Var S:Real;
Begin
S:=0;
For I:=1 To N Do
S:=S+V[I];
Summa:=S;
End;
```

Обращение: `SumD:=Summa(10,D);`

```
Function Summa(M, N:Integer;
V:Matr):Real;
    Var S:Real;
Begin
S:=0;
For I:=1 To M Do
For J:=1 To N Do
S:=S+V[I,J];
Summa:=S;
End;
```

Обращение: `SumB:=Summa(5,5,B);`

Суммирование диагональных элементов.

Для нахождения суммы диагональных элементов (следа) матрицы используется квадратная матрица размера NxN. Блок-схема функции представлена на рис. 11.1.

```
Function Sled(N:Integer;V:Matr):Real;
Var S:Real;
    Begin
       S:=0;
       For I:=1 To N Do
         S:=S+V[I,I];
              Sled:=S;
    End;
```

Обращение из программы: `Slb:=Sled(5,B);`

Суммирование матриц.

Суммой двух матриц A и B называется такая матрица C, элементы которой вычисляются по формуле $c_{ij} = a_{ij} + b_{ij}$ (рис 11.2).

```
Procedure Sum_Matr(M, N:Integer;V,W:Matr;Var U:Matr);
    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(4,5,A,B,C);`

Суммирование элементов матриц по строкам или столбцам.

По строкам

```
Procedure Str(M,N:Integer;
V:Matr; Var St:Mas);
Begin
For I:=1 To M Do Begin
  St[I]:=0;
    For J:=1 To N Do
```

По столбцам

```
Procedure Stlb(M, N: Integer;
V:Matr; Var Stl:Mas);
Begin
For J:=1 To M Do Begin
  Stl[J]:=0;
    For I:=1 To N Do
```

```
    St[I]:=St[I]+V[I,J];
End;
```

Обращение: `Str(4,5,A,D);`

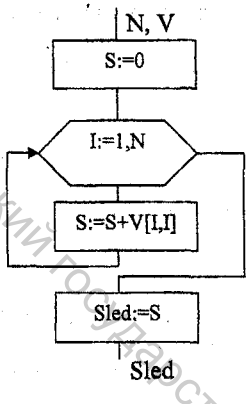


Рис. 11.1. Блок-схема функции вычисления следа матрицы

```
    Stl[J]:=Stl[J]+V[I,J];
End;
```

Обращение: `Stlb(4,5,A,F);`

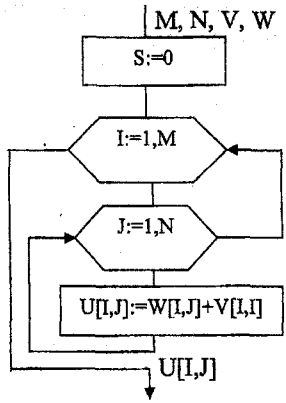


Рис. 11.2 Блок-схема процедуры суммирования матриц

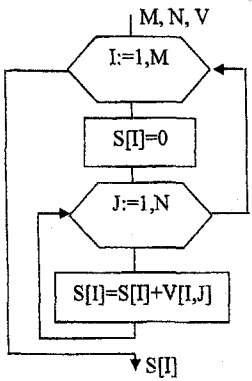


Рис. 11.3. Блок-схема процедуры суммирования элементов по строкам

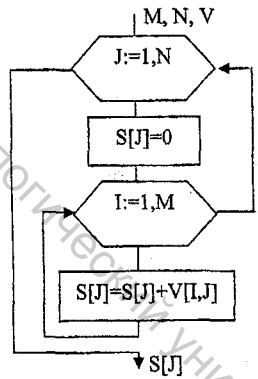


Рис. 11.4. Блок-схема процедуры суммирования элементов по столбцам

Умножение матрицы на вектор.

Результатом умножения матрицы размером MxN на вектор длины N является вектор длины M, компоненты которого вычисляются по формуле $F_i = \sum_{j=1}^{n} A_{ij} \cdot D_j$. Блок-схема процедуры умножения матрицы на вектор представлена на рис. 11.5.

```
Procedure Vector(M,N:Integer;V:Matr;T:Mas;Var S:Mas);
    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);`

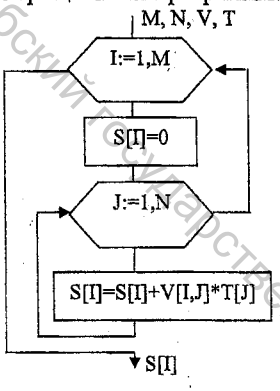


Рис. 11.5. Блок-схема процедуры умножения матрицы на вектор

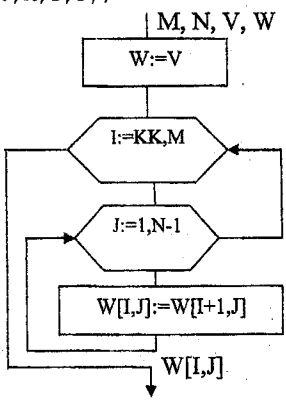


Рис. 11.6. Блок-схема процедуры удаления строки матрицы

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. Блок-схема процедуры представлена на рис. 11.6. Аналогично производится удаление заданного столбца. В этом случае на единицу уменьшается количество столбцов.

```
Procedure Udstr(M,N,KK:Integer;V:Matr;Var W:Matr);
   Begin
     W:=V;
     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);`

Перестановка строк матрицы.

Перестановку строк матрицы можно производить одним из следующих способов: а) с использованием вспомогательной переменной; б) с использованием вспомогательного массива, в который пересылается для временного хранения одна из строк. Блок-схемы представлены на рис. 11.7 и 11.8. Аналогично выполняется перестановка столбцов матрицы.

Пусть требуется поменять местами К-ую и (К+2)-ую строки.

С использованием вспомогательной переменной Р

```
Procedure Per_Str1(N,K:Integer;
V:Matr; Var W:Matr);
Var P:Real;
Begin
W:=V;
For J:=1 To N Do
Begin
P:=W[K,J];
W[K,J]:=W[K+2,J];
W[K+2,J]:=P
End
End;
```

Обращение: `Per_Str1(5,2,A,C);`

С использованием вспомогательного массива Р

```
Procedure Per_Str2(N,K:Integer;
V:Matr; Var W:Matr);
Var P:Mas;
Begin
W:=V;
For J:=1 To N Do
P[J]:=W[K,J];
For J:=1 To N Do
Begin
W[K,J]:=W[K+2,J];
W[K+2,J]:=P[J]
End
End;
```

Обращение: `Per_Str2(5,2,A,C);`

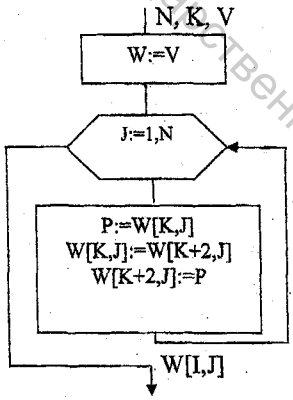


Рис. 11.7. Блок-схема процедуры перестановки строк матрицы с использованием вспомогательной переменной

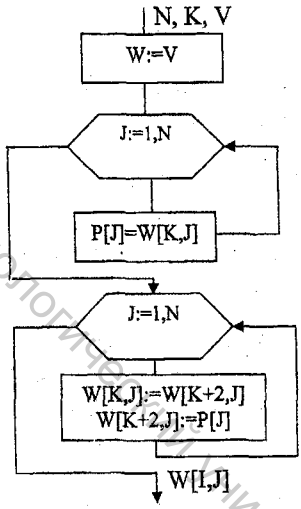


Рис. 11.8. Блок-схема процедуры перестановки строк матрицы с использованием вспомогательного массива

Поиск минимального (максимального) элемента массива.

Рассмотрим принципы поиска на примере поиска максимального элемента в одномерном массиве и поиска минимального элемента в матрице. Результатом поиска может быть значение элемента и/или его порядковый номер (индекс) в массиве. Если необходимо определить только значение элемента, допустимо воспользоваться подпрограммой-функцией. Если, помимо значения

элемента, следует определить и его индекс, необходимо использовать подпрограмму-процедуру.

Поиск максимального элемента в одномерном массиве

```
Function Max(N:Integer;
V:Mas):Real;
Var M:Real;
Begin
   M:=V[1];
     For I:=2 To N Do
        If V[I]>M
          Then M:=V[I];
   Max:=M
End;
```

Обращение: `MaxD:=Max(10,D);`

Поиск минимального элемента в двумерном массиве

```
Procedure Minel (M,N:Integer;
V:Matr; Var Min:Real; Var
K,T:Integer);
Begin
   Min:=V[1,1];
     For I:=1 To M Do
      For J:=1 To N Do
       If V[I,J]<Min Then
         Begin
           Min:=V[I,J];
           K:=I; T:=J
         End
End;
```

Обращение: `Minel(4,5,A,MinA,Q,H);`

Поиск минимального (максимального) элемента строки или столбца.

Результатом этого поиска является одномерный массив длиной M (для строк) или N (для столбцов), поэтому для организации поиска используется подпрограмма-процедура. Блок-схема процедуры представлена на рис. 11.9.

```
Procedure Min_Str(M,N:Integer;
V:Matr; Var Q:Mas);
Begin
  For I:=1 To M Do
   Begin
    Q[I]:=V[I,1];
    For J:=2 To N Do
      If V[I,J]<Q[I]
        Then Q[I]:=V[I,J]
   End
End;
```

Обращение: `Min_Str(4,5,A,D);`

```
Procedure Min_Stlb(M,N:Integer;
V:Matr; Var M_Stl:Mas);
Begin
 For J:=1 To N Do
   Begin
    Q[J]:=V[1,J];
     For I:=1 To M Do
       If V[I,I] <Q[J]
      Then Q[J]:=V[I,J]
   End
End;
```

Обращение: `Min_Stlb(4,5,A,D);`

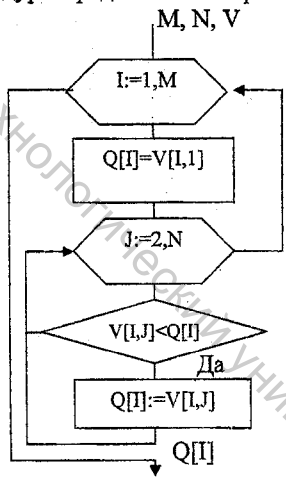


Рис. 11.9. Блок-схема процедуры поиска минимального элемента строки матрицы

11.4. Использование подпрограмм для обработки массивов

Пример 11.3.

Определить средний тарифный разряд рабочих и работ, если известно число разрядов рабочих N=5, число разрядов работ M=3, разряды рабочих R_i={2, 3, 4, 5, 6}; количество рабочих каждого разряда Pi={2, 4, 5, 3, 3}; разряды работ, Si={2, 4, 5}; трудоемкость работ по разрядам, Ti={120, 180, 210}.

Сделать вывод, обеспечен ли участок квалифицированными кадрами.

Средний тарифный разряд рабочих: Средний тарифный разряд работ:

$$RSR = \frac{\sum_{i=1}^{n}(R_i \cdot P_i)}{\sum_{i=1}^{n} P_i} \qquad SSR = \frac{\sum_{j=1}^{m}(S_j \cdot T_j)}{\sum_{j=1}^{m} T_j}$$

Если RSR < SSR, то квалификация рабочих не соответствует квалификации работ.

Текст программы Uchastok на языке Паскаль.

```
Program Uchastok;
 Uses CRT;
 Type mas=array[1..10] of real;
 Var P,R,S,T: mas;
     RSR,SSR: real;
     n,m: integer;
{Процедура ввода элементов одномерного массива}
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;
{Процедура вывода элементов одномерного массива}
 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;
{Функция вычисления суммы элементов одномерного массива}
 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;
{Функция вычисления суммы произведений элементов двух векторов}
  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;
{Текст основной программы}
  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('Массив разрядов работ');
Vyvod(m,S);
Writeln;
Writeln('Трудоемкость работы по каждому разряду');
Vyvod(m,T);
Readln;
End.
```

Результаты работы с программой:

```
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.

Пример 11.4.

В механосборочном цехе завода четыре производственных участка, на которых установлено следующее технологическое оборудование (N_i): токарных станков – 17 шт., фрезерных – 11 шт., протяжных – 14 шт., шлифовальных – 26 шт. Эффективный фонд времени – F=3931 час в год. Среднее выполнение норм выработки – 110% (K=1,1).

Производственная программа выпуска продукции характеризуется следующими данными, приведенными в таблице 11.1. Установить, соответствует ли производственная программа выпуска продукции производственной мощности цеха.

Таблица 11.1. Исходные данные

Показатель	Обозначение	А	Б	В
Количество комплектов на программу выпуска	(Qj)	300	700	500
Плановая трудоемкость токарных работ	T_{ij}	75	35	50
Плановая трудоемкость фрезерных работ		35	25	38
Плановая трудоемкость протяжных работ		40	32	34
Плановая трудоемкость шлифовальных работ		70	80	60

Решение

Вначале определяется станкоемкость планируемой продукции по видам работ по формуле

$$ST_i = \frac{\sum_{j=1}^{n} Q_j \cdot T_{ij}}{K}, \text{ часов/год,}$$

где n – число комплектов оборудования (в данном случае n=3: А, Б, В);

K – коэффициент выполнения норм выработки.

Далее определяется действительный фонд времени по группам оборудования и коэффициент загрузки оборудования.

Действительный фонд времени $FD_i = F \cdot N_i$

Коэффициент загрузки оборудования $KZ_i = \frac{ST_i}{FD_i} \le 1$

Таким образом, исходными данными к задаче являются:

одномерные массивы $N_i = \{17, 11, 14, 26\}$ и $Q_j = \{300, 700, 500\}$,

и матрица трудоемкостей по видам работ $T_{ij} = \begin{vmatrix} 75 & 35 & 50 \\ 35 & 25 & 38 \\ 40 & 32 & 34 \\ 70 & 80 & 60 \end{vmatrix}$.

Ниже приводится программа решения этой задачи на языке Паскаль.

```
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;
{Процедура ввода одномерного массива}
 Procedure Vvod(m:integer;var v:mas);
   Begin
    For i:=1 to m do Readln(v[i]);  End;
{Процедура ввода матрицы}
   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;
{Процедура вывода одномерного массива}
 Procedure Vyvod(m:integer; v:mas);
  Begin
   For i:=1 to m do
    Write(v[i]:5:2,'':2);
    Writeln;
  End;
{Процедура вывода двумерного массива}
  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;
{Текст основной программы}
  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.
```

11.5. 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, тот факт, 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

$$\eta = \frac{y - \bar{y}}{\sigma} \geq t,$$

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 $t \geq 2$, 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).

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, значение минимального, максимального объемов, суммарного выпуска.

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 11.2. Найти валовый выпуск продукции $X_{отч}$ по каждой отрасли.

Таблица 11.2 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 по формуле

$$x_i = \sum_{j=1}^{n} x_{ij} + y_i ,$$

где x_{ij} - межотраслевые потоки продукции;

i и j - соответственно номера отраслей производящих и потребляющих;

y_i - конечная продукция i-той отрасли, $i = 1,\overline{n}$, $j = 1,\overline{n}$.

Вариант 12. Известны межотраслевые потоки продукции 4 отраслей народного хозяйства, представленные в таблице, и валовый выпуск продукции $X_{отч}$ = {292, 250, 193, 254}. Составить матрицу коэффициентов прямых затрат на выпуск продукции по каждой отрасли.

Коэффициент прямых затрат a_{ij} показывает, какое количество продукции i-той отрасли необходимо, учитывая только прямые затраты, для производства единицы продукции j-той отрасли.

Таблица 11.3. 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 является довольно стабильной величиной во времени и вычисляется по формуле

$$a_{ij} = \frac{x_{ij}}{x_j},\ i = 1,\overline{n},\ j = 1,\overline{n},$$

где x_{ij} - межотраслевые потоки продукции;

i и j - соответственно номера отраслей производящих и потребляющих;

x_j - валовый выпуск продукции i-той отраслью.

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.

Длина резки l_i	Количество волокон, n_i шт.
40	2
42	23
44	72
46	3
48	0

Исходную матрицу и значение средней длины резки волокна вывести на экран.

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.

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	Левая чаша	Правая чаша
250	220	230	180
190	220	180	190
200	220	200	175
355	230	245	195
300	245	220	220
235	220	220	175
350	210	260	210
365	225	220	175
190	215	210	175
250	220	250	180

Исходную матрицу, вектора средних значений масс бросков, среднеквадратичных отклонений и коэффициентов вариации по массе вывести на экран.

Вариант 18. В таблице представлено количество часов по декадам, отработанное каждым рабочим раскройного цеха. Определить время, отработанное наименее загруженным рабочим и время, отработанное наиболее загруженным рабочим, а также номера соответствующих 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	62	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.

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

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 вывести на экран.

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

ГЛАВА 12. СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ. СТРОКИ. ЗАПИСИ

12.1. Строка. Описание строкового типа

Как было сказано в предыдущей главе, кроме массивов, к структурированным типам данных относятся строки и записи.

Строка – это последовательность символов кодовой таблицы ASCII компьютера. Для определения данных строкового типа используется идентификатор STRING, за которым следует заключенное в квадратные скобки значение максимально допустимой длины строки данного типа. Количество символов в строке (длина строки) может изменяться от 0 до 255. Если это значение не указывается, то по умолчанию длина строки равна 255 байт [3].

Переменную строкового типа можно определить через описание типа в разделе описания типов или непосредственно в разделе описания переменных.

Формат

1. Через раздел описания типов.

```
Type <имя типа>=string[максимальная длина строки];
Var <идентификатор, …>:<имя типа>;
```

2. Через раздел описания переменных.

```
Var <идентификатор, …>:string[максимальная длина строки];
```

Строковые данные могут использоваться в программе в качестве констант. Значения строкового типа заключаются в апострофы.

Например, `Const Adres='Московский проспект, 72';` – строковая константа, `Name:= 'Иванова Мария Сергеевна';` – переменной Name присвоено значение строкового типа.

Недопустимо применение строковых переменных в качестве селектора в операторе CASE.

Пример

```
Type M_year=string[20];
Var Month: M_year; {описание с заданием типа. Здесь Month – переменная, содержащая название месяца года}
    St1:string; {описание непосредственно в разделе описания переменных, по умолчанию длина строки 255 символов}
    St2,St3:string[25]; {описание непосредственно в разделе описания переменных, максимальная длина строки 25 символов}
    Name:string[265]; {Ошибка! Длина Name превышает 255 символов}
```

В Паскале переменные типа string[N] занимают N+1 байт памяти. Так в приведенном примере строка Month будет занимать 21 байт, St1 – 256 байт, St2 и St3 – по 26 байт. Строка Name не существует, так как ее длина больше 255 символов.

12. 2. Строковые выражения

Выражения, в которых операндами служат строковые данные, называются *строковыми*. Они могут состоять из строковых констант, переменных, указателей функций и знаков операций.

Над строковыми данными допустимы следующие операции:

- сцепления («+») – для сцепления нескольких строк в одну результирующую строку;
- отношения – проводит сравнение двух строковых операндов. Допустимы такие операции отношения как равенство («=»), неравенство («<>»), более чем («>»), менее чем («<»), больше либо равно («>=»), меньше либо равно («<=»).

Вначале выполняются операции сцепления, если они присутствуют, потом реализуются операции отношения. Сравнение производится слева направо до первого несовпадающего символа. Строка считается большей, если ее первый несовпадающий символ имеет больший номер в стандартной таблице обмена информацией. Результат выполнения операций отношения над строковыми операндами всегда имеет булевский тип и принимает значение True, если выражение истинно, и False, если выражение ложно.

Например, условие `'TABLE'<'Table'` будет иметь значение True, так как символ 'A' имеет меньший номер, чем символ 'a'. Условие `'student'>'STUDENT'` также будет истинным, поскольку символ 's' имеет больший номер в таблице ASCII, чем символ 'S'.

Если строки имеют различную длину, но в общей части символы совпадают, считается, что более короткая строка меньше, чем более длинная. Строки считаются равными, если они полностью совпадают по длине и содержат одни и те же символы.

Например, выражение 'WORD2000'<'WORD2000 ' будет истинным, так как длина первой строки меньше второй и составляет 8 и 15 символов соответственно. Результат выражения 'XXXXXX'>'XXX' также будет истинным (True), поскольку длина первой строки равна 6 символам, а второй – 3 символам. Выражение 'Windows'='Windows' принимает значение True, так как все символы имеют одинаковые номера в стандартной таблице обмена информацией, и длина первой строки равна длине второй строки.

Для присваивания строковой переменной результата строкового выражения используется оператор присваивания (:=). Например,

```
Str1:='Студенты первого курса';
Str2:=Str1+'художественно-технологического факультета';
```

Если значение переменной после выполнения оператора присваивания превышает по длине максимально допустимую при описании величину длины строки, все лишние символы отбрасываются.

Пример

```
… … … …
Var Sp1, Sp2: String[25];
       Sp3,Sp4: String[40];
Begin
Sp1:='Технология';
Sp2:='конструирование';
Sp3:=Sp1+' и '+Sp2+' швейных изделий';
Sp4:= Sp1+' и '+Sp2+' из кожи и меха';
… … … …
```

Результатом будут являться строка Sp3, содержащая текст «Технология и конструирование швейных изд», и строка «Технология и конструирование изделий из кожи и м», так как максимально допустимая длина этих строк равна 40 символов. Для исправления этой ошибки следует увеличить длину строки, предположим, до 50 символов.

Допускается смешение в одном выражении операндов строкового и литерного типов. Если при этом литерной переменной присваивается значение строкового типа, длина строки должна быть равна единице, иначе возникает ошибка выполнения.

К отдельным символам строки можно обратиться по номеру (индексу) данного символа в строке. Индекс определяется выражением целочисленного типа, которое записывается в квадратных скобках сразу за идентификатором строковой переменной или константы. Например, `Sp1[2]` обеспечит доступ ко второму символу строки `Sp1`, описанной выше. Этим символом будет буква «е».

Пример 12.1

Найти, сколько раз в списке группы встречаются фамилии с окончанием «ова», например, Иванова, Петрова и т.д.

Решение

Ограничим количество студентов в группе – N=30 человек, и опишем список группы как массив фамилий, в котором переменная i – это номер (индекс) студента в списке группы. Для организации поиска используем цикл для перебора фамилий студентов. Количество символов в фамилии студента будем определять при помощи строковой функции length[3] . В переменной K будем подсчитывать количество студентов, чья фамилия, заканчивается на «ова».

Текст программы на Паскале.

[3] Функция `length(St)` вычисляет длину в символах строки `St`. Результат имеет целочисленный тип.

```
Program Name;
Uses Crt;
Type Group=String[25];
Var Fam:Array[1..30] Of Group;
    i,n,k,j,dl:integer;
Begin
Clrscr;
Writeln('Введите количество студентов в группе');
Readln(n);
For i:=1 to n do
   Begin
     Write('Введите фамилию студента - ');
     Readln(Fam[i])
   End;
Clrscr;
k:=0;
For i:=1 to n do{цикл для организации последовательного перебо-
ра фамилий студентов}
   Begin
      dl:=length(Fam[i]);{Вычисление количества символов в фа-
милии студента}
      If (Fam[i,dl]='a') And (Fam[i,dl-1]='в') And (Fam[i,dl-
2]='о')
      Then k:=k+1
   End;
Writeln('Количество студентов, чья фамилия заканчивается на  -
ова ',k:2);
Writeln('Список группы');
For i:=1 to n do
Writeln(i,' ':3,Fam[i]);
Readln
End.
```

Результат выполнения программы.

```
Количество студентов,чья фамилия заканчивается на -ова, - 4
Список группы
1   Иванов
2   Иванова
3   Петров
4   Петрова
5   Заговалов
6   Повалий
7   Овалов
8   Овалова
```

```
9   Мухова
10  Ковалева
```

12. 3. Записи

При решении прикладных задач часто возникает необходимость обработки таблиц, строки которых содержат поля (столбцы) разного типа данных. Например, технологическая последовательность на обработку изделия содержит такие поля, как № операции, наименование операции, разряд, затрата времени, используемое оборудование, расценка. При этом поля № операции, разряд, затрата времени и расценка содержат числовые значения, а поля наименование операции и используемое оборудование – строковые данные. Такие данные описываются записями.

Запись – это структурированных тип данных, состоящий из фиксированного числа компонентов одного или нескольких типов. Описание этого типа начинается идентификатором Record и заканчивается зарезервированным словом End. Между ними заключен список компонентов, называемых *полями*, с указанием идентификаторов и типа каждого поля [3].

Формат

```
Type <имя типа>=Record
<идентификатор поля>:<тип компонента>;
… … … …
<идентификатор поля>:<тип компонента>
End;
Var <идентификатор, …>:<имя типа>;
```

Пример

Опишем, используя тип запись, студента любой группы.

```
Type Group=record
     N_zach: String[5]; {Номер зачетной книжки}
     FIO: String[40]; {Фамилия, имя, отчество}
     Year: Integer; {Год рождения}
     Adres: String[60]; {Адрес}
     FO: String[2]; {Форма обучения}
     Sr_Ball: Real {Средний балл}
     End;
Var D,Tk:Group;
```

Идентификатор поля должен быть уникален только в пределах записи. Например, идентификатор FIO может использоваться для обозначения фамилии, имени и отчества как в записи Group, так и в других записях, причем длина строки может изменяться. Однако во избежание ошибок идентификатор поля лучше делать уникальным в пределах всей программы.

Объем памяти, необходимый для записи, складывается из длин полей. Для нашего примера необходимый объем памяти равен 121 байт: 6 байт (номер зачетной книжки) плюс 41 байт (поле FIO) плюс 4 байта (поле Year описано типом Integer, значения которого занимают в памяти 4 байта) плюс 61 байт (поле Adres), плюс 3 байта (поле FO), плюс 6 байт (поле Sr_Ball описано типом Real, значения которого занимают в памяти 6 байт).

Доступ к полям записи осуществляется через переменную типа запись. В нашем случае это переменные D и Tk.

Если в задаче используется несколько одинаковых переменных типа запись, то удобно воспользоваться массивами из записей. Формат описания массивов такого типа соответствует описанию данных структурированного типа массив и рассмотрен в главе 11.

Пример

Опишем бригаду рабочих раскройного цеха.

```
Type Br=record
     Tab_N: Integer; {табельный номер рабочего}
     FIO: String[40]; {фамилия, имя, отчество рабочего}
     Spec: String[30]; {специальность}
     N_vyrabotki: Byte; {норма выработки}
     F_vyrabotka: Byte; {фактическая выработка}
     KVN: Real {коэффициент выполнения нормы}
     End;
Var Brigada: Array[1..20] Of Group; {бригада рабочих раскройного цеха, не более 20 человек}
```

Значения полей записи могут быть использованы в выражениях. Так как существует требование к уникальности идентификаторов полей записи только внутри типа, один идентификатор может использоваться в нескольких записях. Например, идентификатор FIO используется в записи Group для обозначения фамилии, имени и отчества студента, а в записи Br – для идентификации рабочего бригады. Поэтому к именам отдельных полей нельзя обращаться по аналогии с идентификаторами переменных, например, Adres, FIO, Year. Обращение к значению поля осуществляется с помощью идентификатора переменной и идентификатора поля, разделенных точкой. Такая комбинация называется *составным именем*. Обращение к полям записи Group в этом случае выглядит так:

```
Group.N_zach, Group.FIO, Group.Year, Group.Adres
```

Составное имя можно использовать везде, где допустимо применение типа данных поля. Для присваивания полям значений используется оператор присваивания. Если две записи описаны одинаково, то операцию присваивания допустимо выполнять применительно к записям в целом. Например, Tk:=D; После выполне-

ния этого оператора значения полей записи Tk станут равны значениям соответствующих полей записи D.

Пример

```
Group.N_zach:='05672';
Group.FIO:='Сергеева Марина Антоновна';
Group.Year:=1986;
```

Составные имена можно использовать в операторах ввода-вывода. Например, `Read(Group.N_zach, Group.FIO, Group.Year);` или `Write(Group.N_zach, Group.FIO, Group.Year:4);`

Однако подобное обращение к полям записи имеет несколько громоздкий вид. Для решения этой проблемы в языке Паскаль существует оператор WITH, который имеет следующий формат:

```
WITH <переменная типа запись> DO <оператор>;
```

Указав переменную типа запись в операторе WITH один раз, имена полей можно использовать как обычные переменные, т.е. без указания перед идентификатором поля имени переменной, определяющей запись.

Пример

1) Присвоить значения полям записи Group с помощью оператора WITH.

```
With Group Do
Begin
N_zach:='05672';
FIO:='Сергеева Марина Антоновна';
Year:=1986
End;
```

2) Задать значения для второго рабочего бригады раскройного цеха с помощью оператора WITH.

```
With Brigada[2] Do
Begin
Tab_N:=123;
FIO:='Никонов Василий Михайлович';
Spec:='закройщик';
N_vyrabotki:=35;
F_vyrabotka:=37
End;
```

Поле KVN рассчитывается в зависимости от нормы выработки и ее фактического выполнения, поэтому значение для этого поля не указывается.

Паскаль допускает вложение записей друг в друга, поэтому оператор WITH может быть вложенным. Уровень вложений не должен превышать 9.

```
With Rv1 Do
     With Rv2 Do
          ...
          With RvN Do
```

Такая конструкция эквивалента конструкции
```
With Rv1,Rv2,Rv3, .., RvN Do ..
```

12.4. Использование строк и записей при решении прикладных задач

Пример 12.2

На поточной линии выпускаются валики определенного диаметра. Для их изготовления выполняются операции, представленные в таблице 12.1. Трудоемкость выполнения каждой операции (T_i, мин) также представлена в таблице 12.1.

Определить расчетное (W_p, чел.) и фактическое ($W_ф$, чел.) количество рабочих мест по каждой операции и их загрузку ($K_{з.о.i}$, %), если рабочий такт линии Ч=1,66 мин. Найти суммарную трудоемкость, общее расчетное и фактическое количество рабочих.

Расчетное количество рабочих	Фактическое количество рабочих	Коэффициент загрузки рабочих мест
$W_p = \frac{T_i}{Ч}$	$W_ф$= округлить до целого W_p	$K_{з.о.i} = \frac{W_p}{W_ф} \cdot 100$

Таблица 12.1. Исходные данные

Наименование операции	T_i	W_p	$W_ф$	$K_{з.о.i}$, %
Токарная	1,5			
Токарная	6,0			
Фрезерная	4,3			
Шлифовальная	2,5			

Решение

Так как рабочий такт линии известен из условия и в процессе выполнения программы не изменяется, опишем его как константу Ch. Алгоритм решения задачи можно представить в виде последовательности следующих действий:

1. Ввод исходных данных.
2. Проведение расчетов (вычисление W_p, $W_ф$, $K_{з.о.i}$, суммарной трудоемкости, общего расчетного и фактического количества рабочих).
3. Вывод таблицы на экран.
4. Вывод итоговых значений (суммарной трудоемкости, общего расчетного и фактического количества рабочих).

Текст программы на Паскале.

```
Program Valiki;
Uses Crt;
```

```
    Const Ch=1.66; {Рабочий такт линии}
    Type Tabl=RECORD {Описание типа запись}
         Oper:String;
         Ti:Real;
         Wr:Real;
         Wf:Byte;
         Kzoi:Real
         END;
    Var R_m:Array[1..4] Of Tabl; {массив переменных типа Tabl}
        I,SWf:Byte; {параметр цикла, общее фактическое количество рабочих}
        STi,SWr:Real; {суммарная трудоемкость и общее расчетное количество рабочих}
    Begin
    Clrscr;
      Writeln('Ввод исходных данных');
  For i:=1 to 4 do {Организация цикла для ввода исходных данных}
   With R_m[i] do  {подключение переменной типа RECORD}
    Begin
     Write('Введите наименование операции => ');
      Readln(Oper);{ввод наименования операции}
     Write('Введите трудоемкость => ');
      Readln(Ti)   {ввод трудоемкости}
    End;
 STi:=0; SWr:=0; SWf:=0; {Проведение расчетов}
  For i:=1 to 4 do
   With R_m[i] do
    Begin
     STi:=STi+Ti;       {вычисление суммарной трудоемкости}
     Wr:=Ti/Ch;         {вычисление Wr}
     SWr:=SWr+Wr;       {вычисление общего расчетного количества рабочих}
     Wf:=Round(Wr);     {вычисление Wf}
     SWf:=SWf+Wf;       {вычисление общего фактического количества рабочих}
     Kzoi:=Wr/Wf*100    {вычисление Kzoi}
    End;
  Writeln('Операция        Ti      Wr      Wf      Kzoi'); {Вывод заголовков столбцов}
   For i:=1 to 4 do
    With R_m[i] do
     Writeln(Oper,' ':3,Ti:3:1,' ':3,Wr:6:3,' ':3,' ',Wf,'
':3,Kzoi:5:2);
   Writeln('Суммарная трудоемкость - ',Sti:4:1);
   Writeln('Общее расчетное количество рабочих - ',SWr:4:1);
   Writeln ('Общее фактическое количество рабочих - ',Swf);
 Readln
```

```
End.
```

Результат работы программы.

```
  Ввод исходных данных
  Введите наименование операции => Токарная
  Введите трудоемкость => 1.5
  Введите наименование операции => Токарная
  Введите трудоемкость => 6
  Введите наименование операции => Фрезерная
  Введите трудоемкость => 4.3
  Введите наименование операции => Шлифовальная
  Введите трудоемкость => 2.5
  Операция        Ti      Wr      Wf      Kzoi
  Токарная    1.5     0.904     1    90.36
  Токарная    6.0     3.614     4    90.36
  Фрезерная   4.3     2.590     3    86.35
  Шлифовальная    2.5     1.506     2    75.30
  Суммарная трудоемкость - 14.3
  Общее расчетное количество рабочих -  8.6
  Общее фактическое количество рабочих - 10
```

12.5. Задания для самостоятельной работы

Вариант 1.

Определить основные свойства волокон, представленные в таблице 12.2.

Таблица 12.2. Основные свойства химических волокон новой структуры

Волокно	Линейная плотность, T, текс	Средняя плотность, γ, г/см^3	Диаметр волокна, мкм	Метрический номер	Разрывная нагрузка, сН	Разрывная длина, мм
Лавсан	0,33	1,38				
Нитрон Н-4	0,33	1,17				
Нитрон НС	0,56	1,17				
Канекарон RMK	1,2	1,17				
Канекарон UHD	0,33	1,17				
Канекарон FHS	0,22	1,17				

Диаметр волокна, мкм	Метрический номер	Разрывная нагрузка волокна, сН	Разрывная длина волокна, мм
$d = 35{,}7\sqrt{\frac{T}{\gamma}}$	$N = \frac{957000}{d^2}$	$P = 0{,}0056d^2 + 0{,}503d - 3{,}77$	$L = 5{,}6 + 502/d - 3790/d^3$

Вариант 2.

В таблице 12.3 представлены сведения о выпуске ткани за 12 месяцев 200х года, тыс. погонных метров. Определить расчетное значение выпуска продукции y_p и отклонение фактического выпуска от расчетного, %. Для определения y_p воспользуйтесь уравнением $y_p = a \cdot x + b$, где a и b – коэффициенты уравнения, вычисляемые по формулам

a	b	Отклонение фактического выпуска от расчетного, %
$a = \frac{n \cdot \sum_{i=1}^{n} y_i \cdot x_i - \sum_{i=1}^{n} y_i \cdot \sum_{i=1}^{n} x_i}{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$	$b = \frac{\sum_{i=1}^{n} y_i \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \cdot \sum_{i=1}^{n} x_i}{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$	$Otkl = \frac{y_{\phi_i} - y_{p_i}}{y_{\phi_i}} \cdot 100$

Здесь x_i – номер месяца (i=1..12), y_i – фактический выпуск продукции, n - количество месяцев (n=12), $\sum_{i=1}^{n} x_i$, $\sum_{i=1}^{n} y_i$, $\sum_{i=1}^{n} y_i \cdot x_i$, $\sum_{i=1}^{n} x_i^2$ - суммы столбцов 2-5 соответственно.

Таблица 12.3. Выпуск ткани за 12 месяцев 200х года, тыс. п.м.

Месяц	Номер месяца, x	Фактический выпуск продукции, тыс. п.м., y_Φ	$x_i \cdot y_{\phi_i}$	x_i^2	Расчетное значение выпуска продукции, тыс. п.м., y_p	Отклонение фактического выпуска от расчетного, %
1	*2*	*3*	*4*	*5*	*6*	*7*
Январь	1	257				
Февраль	2	254,1				
Март	3	251,2				
Апрель	4	248,4				
Май	5	245,5				
Июнь	6	242,6				
Июль	7	236,3				
Август	8	239,6				
Сентябрь	9	235,7				
Октябрь	10	231,8				
Ноябрь	11	227,8				
Декабрь	12	223,9				

Вариант 3. В таблице 12.4 представлены данные о деформационных свойствах обувных материалов.

Таблица 12.4. Данные о деформационных свойствах обувных материалов

Наименование материала и направление раскроя	P_0, Н	P_1, Н	P_2, Н	$P_{рел.}$, Н	$P_б$, %	$P_м$, %	$P_{общ.}$, %	$P_з$, %	$K_{пад}$
Натуральная кожа (НК) вдоль	446,78	383,61	315,84						
НК поперек	163,18	138,18	111,86						
Трикотаж плотностью 170 г/м^2	5,43	4,84	3,54						
НК вдоль + трикотаж (пл. 170 г/м^2)	476,36	402,76	230,29						
НК поперек + трикотаж (пл. 170 г/м^2)	177,66	147,35	96,73						

Определить релаксируемое усилие $P_{рел.}$, доли быстропротекающих $P_б$, медленнопротекающих $P_м$, заторможенных процессов релаксации $P_з$, общую долю релаксации усилия $P_{общ.}$ и коэффициент падения усилия $K_{пад}$. Найти средние значения рассчитываемых показателей ($P_{рел.}$, $P_б$, $P_м$, $P_з$, $P_{общ.}$ и $K_{пад}$).

$P_{рел} = P_0 - P_2$	$P_б = \frac{P_0 - P_1}{P_0} \cdot 100$	$P_м = \frac{P_1 - P_2}{P_0} \cdot 100$
$P_з = \frac{P_2}{P_0} \cdot 100$	$P_{общ} = \frac{P_0 - P_2}{P_0} \cdot 100$	$K_{пад} = \frac{P_0}{P_2}$

Вариант 4. В таблице 12.5 представлены данные о полотнах тканей различных артикулов. Определить нормировочные коэффициенты l и n, комплексный показатель эффективности использования данных полотен, средние значения длины полотна и нормы отходов на настил, а также минимальные значения нормировочных коэффициентов l и n.

Нормировочные коэффициенты		Комплексный показатель эффективности
$l = \frac{L}{L_{max}}$	$n = \frac{H}{H_{max}}$	$K = \frac{1}{2}\left(\frac{1-l}{1-l_{min}} + \frac{1-n}{1-n_{min}}\right)$

Здесь L и H – длина полотна и норма отходов на настил, м, L_{max} и H_{max} – максимальные длина и норма отходов по длине, м., l_{min} и n_{min} – минимальные значения коэффициентов l и n соответственно.

Таблица 12.5. Сведения о полотнах тканей различных артикулов

Артикул ткани	Длина полотна, L, м	Максимальная длина полотна в настиле, L_{max}, м	Норма отходов на настил, H, м	Максимальное значение нормы отходов по длине, H_{max}, м	*l*	n	Комплексный показатель эффективности
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
8c71ТЯ	4,35	6,69	0,0152	0,023			
8c71ТЯ	4,67	6,76	0,0157	0,023			
NICOLE	5,78	8,89	0,0148	0,018			
NICOLE	5,78	8,5	0,0154	0,018			
VELOUR	4,1	7,32	0,0126	0,021			
69404/71	5,13	6,4	0,018	0,025			
69404/71	4,74	6,32	0,017	0,025			

Вариант 5. Рассчитать основные технико-экономические показатели (ТЭП) швейного потока по изготовлению женских блузок, представленные в таблице 12.6. Мощность потока в смену, M – 360 ед/см. Время смены, R – 28800 с.

Таблица 12.6. ТЭП швейного потока по изготовлению женских блузок

Модель	Трудоемкость модели, T_i	Соответствие выпуска по моделям, mi	Выпуск по моделям в смену, Mi	Общая трудоемкость по выпуску, Ot_i	Удельная трудоемкость, g_i	Время выполнения сменного задания, Ri	Такт по моделям, ti	Время выполнения по моделям	Расчетное количество рабочих по моделям, Np
А	5574	1							
Б	5496	2							
В	5544	1							
Итого									

$M_i = \frac{M}{\Sigma m_i} \cdot m_i$ Округлить до целого	$Ot_i = T_i \cdot M_i$	$\gamma_i = \frac{T_i \cdot M_i}{\Sigma T_i \cdot M_i}$	$R_i = \gamma_i * R$
$t_i = \frac{R_i}{M_i}$	$M_i^{ч} = \frac{3600}{t_i}$	$N_p = \frac{T_i}{t_i}$	

Вариант 6. Рассчитать параметры заправки автоматических весовых питателей, используя исходные данные, представленные в таблице 12.7. Общая масса

броска составляет 5 кг. Время цикла – 24 с. Найти максимальное время наполнения чаши весов, с, и минимальную скорость наклонной игольчатой решетки.

Таблица 12.7. Параметры заправки автоматических весовых питателей

№ питателя	Вид волокна	Масса броска, г	% вложения	Время наполнения чаши весов, с	Время наполнения чаши весов, %	Скорость наклонной игольчатой решетки, v, м/мин	Частота вращения наклонной игольчатой решетки, n, $мин^{-1}$	Скорость питающей решетки, v, м/мин	Частота вращения питающей решетки, n, $мин^{-1}$
№1	Лавсан	1000		8		26			5
№2	Нитрон Н-4	750		16		21			4,3
№3	Лавсан	750		18		21			4,6
№4	Нитрон Н-4	750		9		22			4,3
№5	Нитрон Н-2	1000		15		24			4,6
№6	Лавсан	750		16		17			3,2

% вложения = Масса броска / Общая масса броска * 100.

Время наполнения чаши весов, %, = Время наполнения чаши весов / Время цикла * 100.

Частота вращения наклонной игольчатой решетки = скорость наклонной игольчатой решетки / π * d_1, где d_1 – диаметр наклонной игольчатой решетки, d_1=0,155 м.

Скорость питающей решетки = частота вращения питающей решетки*π*d_2, где d_2 – диаметр питающей решетки, d_2=0,155 м.

Вариант 7. Ведомость выпуска изделий текстильным предприятием содержит поля: код цеха, цех, код изделия, наименование изделия, выпуск по плану, фактический выпуск, плановая себестоимость, фактическая себестоимость, *сумма по плану*, сумма по факту**. Ввести не менее 10 – 15 значений записей. Поля, помеченные знаком «*», оставить незаполненными.

Сумма по плану = выпуск по плану * плановая себестоимость.

Сумма по факту = фактический выпуск * фактическая себестоимость.

Найти общее плановое количество, среднюю фактическую себестоимость, минимальную сумму по плану, максимальную сумму по факту по каждому цеху.

Найти и вывести на экран название цеха, выпустившего максимальное количество продукции.

Вариант 8. В таблице 12.8 представлены сведения о длинах кусков плащевой ткани артикула 3С37ТЯ. Норма на обмелку комплекта изделий размеро-роста 170-96 составляет 3,85 м, 176-100 – 4,06 м, 182-100 – 4,2 м.

Таблица 12.8. Длины кусков

Фабричный номер куска	Длина куска ткани, м	Количество комплектов размеро-роста			Длина концевого остатка, м			Максимальное количество комплектов	Максимальная длина остатка, м
		170-96	176-100	182-100	170-96	176-100	182-100		
8801	34,9								
8724	35,1								
8736	33,35								
8782	33,6								
8800	33,1								
8795	32,5								
8668	31,6								
8804	32,2								
8727	33,9								
8680	33								

Количество комплектов = Длина куска / Норма на обмелку. Округлить до ближайшего меньшего целого.

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

По каждому размеро-росту определить количество комплектов изделий, длины концевых остатков после раскроя куска а также средние длины концевых остатков.

Для каждого куска найти максимальное количество комплектов и максимальную длину концевого остатка.

Найти среднюю длину куска ткани.

Вариант 9. Бригадная ведомость рабочих текстильного предприятия содержит следующие сведения: табельный №, ФИО, специальность, разряд, тарифная ставка, количество отработанных часов, *тарифный заработок*, сдельный заработок**. Ввести не менее 10 – 15 значений записей. Поля, помеченные знаком «*», оставить незаполненными.

Тарифный заработок = тарифная ставка * количество отработанных часов.

Сдельный заработок = тарифный заработок * КТУ, причем КТУ устанавливается в зависимости от количества отработанного времени: от 0 до 150 часов – 1,2, свыше 150 часов – 1,5.

Вывести на экран сведения о работниках, чья фамилия начинается с определенной буквы, тарифная ставка которых лежит в заданных пределах. Значения тарифной ставки задать самостоятельно.

Вариант 10. В списке работников предприятия указаны данные: табельный №, ФИО, семейное положение, количество детей, % надбавки, тарифный оклад, *размер оклада с учетом надбавки**, *размер пособия на детей**. Ввести не менее 10 – 15 значений записей. Поля, помеченные знаком «*», оставить незаполненными.

Размер оклада с учетом надбавки = (1+% надбавки/100)*оклад.

Обновляемое поле размер пособия на детей* рассчитывается в зависимости от количества детей следующим образом: если количество детей 1-2, размер пособия на детей = 0,1 * размер оклада * количество детей, если более 3, то размер пособия на детей = 0,2 * размер оклада * количество детей.

Найти количество работников, имеющих одинаковое семейное положение и средний размер их тарифного оклада.

ГЛАВА 13. ПОИСК И СОРТИРОВКА

Упорядочивание элементов совокупности позволяет упростить дальнейший поиск или представить полученные результаты в более понятном виде. Рассмотрим основные определения.

Сортировкой называется процесс упорядочивания набора данных одного типа по возрастанию или убыванию значения какого-либо признака [3].

Поиск – это просмотр некоторого набора однотипных данных с целью нахождения одного и более элементов, обладающих определенным свойством или с целью установления факта отсутствия таких элементов [3].

Ранее были рассмотрены методы поиска минимального или максимального элементов в одномерном или двумерном массивах. В этом случае решением задачи являлось либо значение элемента массива, либо его индекс в массиве, либо и то, и другое.

В этой главе будут рассмотрены методы поиска элемента, равного некоторому заранее заданному `X`, называемому *ключом поиска*, в группе данных, представляющей собой одномерную таблицу `Q` из `N` элементов. Полученный в результате индекс `i`, удовлетворяющий условию `A[i] = x`, считается решением задачи.

Все методы поиска подразделяются на две группы: поиск в упорядоченном наборе и поиск в неупорядоченном наборе данных.

Так как процесс упорядочивания элементов массива позволяет упростить поиск в нем, рассмотрим основные алгоритмы сортировки массива.

13.1. Сортировка массивов

Массив, например, массив $A=\{a_1, a_2, \ldots, a_n\}$, является отсортированным (упорядоченным) по возрастанию, если для всех индексов `i` из интервала $i=\overline{1,n+1}$ выполняется условие $a_i \le a_{i+1}$.

Сортировка строк – это ранжирование данных строкового типа по алфавиту. Фактически символы, составляющие строку, упорядочиваются по их номеру в кодовой таблице ASCII. Наиболее часто сортировку строк выполняют при выводе различных списков (фамилий, названий). Упорядочивать строки можно любым из рассматриваемых в данной главе методов.

Различают обменные сортировки, сортировки выбором, сортировку вставками, сортировку слиянием. Рассмотрим некоторые методы данных видов сортировок. Предположим, что имеется исходный массив $A=\{a_1, a_2, \ldots, a_{10}\}$, который необходимо упорядочить возрастанию.

Обменные сортировки

Суть обменных сортировок заключается в следующем. Сначала последовательно просматриваются элементы $a_1, \ldots, a_n$ и находится наименьшее i, такое, что $a_i > a_{i+1}$. Затем элементы a_i и a_{i+1} меняются местами, и просмотр возобновляется с a_{i+1} элемента, и т.д. 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. Процесс сортировки по убыванию будет аналогичен.

В пределах этого вида можно выделить такие методы сортировок, как метод «пузырька»; метод улучшенной «пузырьковой» сортировки; метод быстрой сортировки.

Для сортировки *методом «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 13.1

Пусть задан массив A, состоящий из элементов: `A={25; 57; 48; 37; 12; 92; 86; 33}`. Упорядочить его по возрастанию значений элементов.

Решение

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 таком 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, словно пузырек, медленно всплывает 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 A 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 основан на методе «пузырька» с той лишь разницей, что количество итераций меньше. Рассмотрим его на примере.

Пример 13.2

Упорядочим по возрастанию методом «пузырька» массив A, представленный в примере 13.1, а также проиллюстрируем метод улучшенной «пузырьковой» сортировки на примере упорядочивания данного массива по убыванию.

Решение

Создаем вспомогательные массивы B и C. В массиве B будут сохраняться результаты сортировки по убыванию, в массиве C – по возрастанию. Исходный массив A и отсортированные массивы B и C выведем на экран, используя подпрограмму-процедуру.

```
Program Sort1;
Uses CRT;
Type Mas=Array[1..50] Of Integer;
Var A,B,C:Mas;
    i,j,p,N:integer;
{Процедура вывода элементов массива}
Procedure Vyvod(M:Integer; Var V:Mas);
Begin
  For i:=1 to M do
    Write(V[i]:2,' ':2);
  Writeln
End;
Begin
Writeln('Введите количество элементов массива A');
Readln(N);
Writeln('Введите массив A');
 For i:=1 to N do
    Readln(A[i])
Writeln('Исходный массив');
Vyvod(N,A);
{Сортировка массива по возрастанию методом «пузырька»}
C:=A;
For i:=1 to N do
  For j:=1 to N-1 do
    If C[j]>C[j+1] Then
```

```
        Begin
          P:=C[j];
          C[j]:=C[j+1];
          C[j+1]:=P
        End;
Writeln('Отсортированный по возрастанию массив A');
Vyvod(N,C);
```

{Сортировка массива по убыванию методом улучшенной «пузырьковой» сортировки. При использовании метода улучшенной «пузырьковой» сортировки i изменяется от 1 до n-1, j от 1 до n-i}

```
B:=A;
For i:=1 to N-1 do
  For j:=1 to N-i do
    If B[j]<B[j+1] Then
      Begin
        P:=B[j];
        B[j]:=B[j+1];
        B[j+1]:=P
      End;
Writeln('Отсортированный по убыванию массив A');
Vyvod(N,B);
Readln
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

Метод *быстрой сортировки* является еще одной реализацией обменных сортировок. Пусть имеется массив A, состоящий из N элементов. Для обозначения индексов элементов воспользуемся двумя переменными – i и j. Полагаем i=1, j=n. Если a_i меньше a_j, то j уменьшаем на единицу и снова сравниваем a_i и a_j. После первой перестановки i увеличивается на единицу, и выполняется еще одна перестановка. Затем j уменьшаем на единицу. Процесс сравнения продолжается до тех пор, пока исходный набор не будет упорядочен по возрастанию.

Рассмотрим этот метод на примере сортировки строк.

Пример 13.3

Имеется список бригады рабочих из 10 человек. Упорядочить его по возрастанию методом быстрой сортировки и вывести на экран.

Решение

Обозначим через переменную w строку длиной не более 20 символов (фамилия рабочего), v – массив из 10 элементов (количество рабочих), p – вспомогательная переменная строкового типа для перестановки элементов, i и j – индексы строк в массиве. Программа на языке Паскаль представлена ниже.

```
Program Sort_String;
Uses Crt;
Type W=String[20];
     V=Array[1..10] Of W;
Var Fio:V;
    I,J:Integer;
    P:W;
Begin
Clrscr;
Writeln('Ввод списка бригады');
For I:=1 To 10 Do
 Begin
   Write(i,' рабочий - ');
     Readln(FIO[I])
 End;
For I:=1 To 10 Do {Метод быстрой сортировки}
  For J:=10 Downto 1 Do
   If (FIO[I]>FIO[J]) And (I<=J) Then
     Begin
       P:=FIO[I];
       FIO[I]:=FIO[J];
       FIO[J]:=P;
     End;
Writeln('Отсортированный список бригады');
For I:=1 To 10 Do
  Writeln(Fio[I]);
Readln
End.
```

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 13.4

Отсортировать по возрастанию массив A из элементов 1 3 0 9 2.

Решение

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 программа, реализующая данный метод.

```
Program Sort2;
 Uses Crt;
  Var A: Array[1..50] Of Real;
      I,J,N,K: Integer;
      Min,W: Real;
{Процедура вывода элементов массива}
Procedure Vyvod(M:Integer; Var V:Mas);
Begin
  For i:=1 to M do
    Write(V[i]:2,' ':2);
  Writeln
End;
 Begin
     Clrscr;
     Writeln('Введите количество элементов массива A');
     Readln(n);
     Writeln('Введите массив A');
     For i:=1 to n do
     Read(a[i]);
     Writeln;
     Writeln('Исходный массив A');
     Vyvod(N, A);
   {Сортировка массива выбором}
     For i:=1 to n do
      Begin
       min:=a[i]; k:=i;
       For j:=i+1 to n 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');
        Vyvod(N, A);
        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

Аналогичным данному методу и одним из самых простых методов сортировки является метод *прямого выбора*. Для его реализации выбирается максимальный элемент. Выбранный элемент меняется местами с последним элементом; процесс повторяется с оставшимися n-1, n-2 элементами и т.д., пока не останется один, самый малый по значению элемент.

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 переборе массива $a_2, \ldots, a_n$ с тем, чтобы каждый новый элемент a_i вставлять на подходящее место в уже упорядоченной совокупности $a_1, \ldots, a_{i-1}$. Это место определяется последовательным сравнением a_i с упорядоченными элементами $a_1, \ldots, a_{i-1}$.

При сортировке «*бинарными*» вставками место, на которое надо вставить элемент a_i в уже упорядоченную совокупность $a_1, \ldots, a_{i-1}$, определяется алгоритмом деления пополам. Таким образом, получается алгоритм сортировки «*бинарными*» вставками, который понимается как «вставка делением пополам».

Пример 13.5

Вставить в упорядоченный по возрастанию массив A = {3; 4; 7; 9} новый элемент, равный 5, таким образом, чтобы сохранилась упорядоченность.

Решение

Алгоритм решения задачи следующий:

1. Найти в массиве элемент, больший вставляемого. Для этого следует просмотреть все элементы, начиная с первого. Это элемент a[3] = 7.

2. Увеличить длину массива на 1. Результат – массив A = {3 4 7 9 X}.

3. Все элементы, стоящие справа от найденного, включая его самого, то есть от 3-го, сдвинуть вправо. Результат – 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,n,k: integer;
      g:real;
{Процедура вывода элементов массива}
Procedure Vyvod(M:Integer; Var V:Mas);
Begin
  For i:=1 to M do
    Write(V[i]:2,' ':2);
  Writeln
End;
 Begin
     Clrscr;
     Writeln('Введите количество элементов массива A');
     Readln(n);
     Writeln('Введите массив A');
     For i:=1 to n do
     Read(a[i]);
     Writeln;
     Writeln('Введите число, которое нужно вставить');
     Readln(g);
     Writeln('Исходный массив A');
     Vyvod(N, A);
     {1. Ищем элемент больше вставляемого}
     k:=1; {k - индекс сравниваемого элемента}
     While (k<=n) and (g>=a[k]) do {если k не превышает n и
вставляемый элемент меньше или равен a[k]}
         k:=k+1;{то переходим к следующему элементу}
     {2. Увеличиваем длину массива на 1}
     n:=n+1;
     {3.Сдвигаем элементы, начиная с k-го вправо}
     For i:=n downto k+1 do
```

```
        a[i]:=a[i-1];
      {4.В a[k] заносим g}
      a[k]:=g;
      Writeln('Новый массив A');
      Vyvod(N, A);
      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

Сортировка «слияниями» (Алгоритм фон Неймана)

Суть заключается в упорядочении массива $a_1, \ldots, a_n$ по неубыванию. Метод основан на многократных слияниях уже упорядоченных групп элементов массива. Сначала 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 использовать вспомогательный массив $b_1, \ldots, b_n$.

13.2. Поиск в неупорядоченном наборе данных

В главе 11 уже рассматривались методы поиска минимального или максимального элемента в массиве. При этом полагали, что минимальным (максимальным) элементом сначала является первый элемент. Затем последовательно просматривали элементы массива и сравнивали их со значением минимума (максимума). Если элемент оказывался меньше (больше) минимума (максимума), то полагали минимальным (максимальным) данный элемент. Такая реализация является частным случаем метода *линейного* (последовательного) поиска, суть которого заключается в последовательном переборе элементов массива и сравнении их значений с ключом. Разумеется, искомого значения в массиве может и не быть. При достаточно большом объеме совокупности процесс поиска может затянуться, поэтому для решения таких задач следует сначала отсор-

тировать совокупность, а затем производить поиск в уже упорядоченном наборе данных. Рассмотрим реализацию данного метода на следующем примере.

Пример 13.6

В строке А произвольной длины найти индекс элемента, равного 'а'. Если элемент не обнаружен, вернуть значение 0.

```
Program Stroka;
Uses Crt;
Type Str=String;
Var S: Str;
    I, J, K:Integer;
Begin
Clrscr;
  Writeln('Введите строку');
    Readln(S);
  K:=0;
  For I:=1 To 255 Do
    If S[I]='a' Then
      Begin
        Writeln('Индекс элемента - ',I:3);
        K:=K+1
      End;
  If K=0 Then Writeln(K:2);
Readln
End.
```

13.3. Поиск в упорядоченном наборе данных

Одним из методов поиска в упорядоченном наборе данных является *бинарный* поиск, суть которого в следующем. Сначала находим средний элемент совокупности $A_{N/2}$, затем сравниваем его с ключом K. Если $A_{N/2}$ меньше значения ключа, то дальнейший поиск организуется среди элементов с номерами от N/2 до N, в противном случае – от 1 до N/2. В полученной группе элементов опять находим среднее и выполняем аналогичное сравнение. Процесс продолжаем до тех пор, пока не будет найден искомый элемент. Для получения конечного результата требуется в среднем $\log_2 N$ шагов.

Пример 13.7

В отсортированном массиве A={8; 10; 44; 56; 6; 16; 23; 18; 65; 12} найти индекс элемента, равного 6. Если элемент не обнаружен, вывести соответствующее сообщение.

Решение

Определим переменные. A – исходный массив, Sr – значение среднего элемента массива A, K – значение ключа, T – нижняя граница интервала, Q – верхняя граница интервала, переменная B используется для определения нали-

чия элемента, равного ключу в данном массиве. Сначала переменная B имеет значение False. В случае, если в массиве присутствует нужный элемент, переменная B становится равной True. Текст программы представлен ниже.

```
Program Search;
Const N=10;
Type Mas=Array[1..N] Of Integer;
Var A:Mas;
    i,j,T,Q:Byte;
    Sr,K:Integer;
    B:Boolean;
Begin
For i:=1 to N Do
 Begin
   Write(i,' элемент - ');
   Readln(A[i])
 End;
Writeln('Введите значение ключа');
Readln(K);
T:=1; Q:=N;
B:=False;
For i:=1 to N Do
 Begin
   Sr:=(A[T]+A[Q]) Div 2;
   If A[i]=K Then Begin B:=True;
Writeln('Элемент, равный ',K, 'находится под номером', I)
   End
   Else If A[i]>K Then Q:=N Div 2 Else T:=N Div 2
 End;
Writeln('Исходный массив A');
For i:=1 to N do
   Write(A[i]:2,' ');
Writeln;
If B=False Then Writeln('Такого элемента нет');
Readln
End.
```

13.4. Задания для самостоятельной работы

Задания для самостоятельной работы основаны на задачах, изложенных в главах 11 и 12. Предлагаемые варианты заданий можно решать как отдельные задачи для закрепления темы «Сортировка и поиск», либо использовать их как возможность усложнения задач по теме «Структурированные типы данных. Массивы. Строки. Записи».

Вариант 1. Для условия задачи варианта 2 главы 11 отсортировать по убыванию значения вектора количества работников, имеющих надбавку за профессионализм. Методом бинарного поиска найти количество подразделений, в которых работники не получают надбавку.

Вариант 2. Для условия задачи варианта 6 главы 11 определить, сколько раз в векторе выпуска трикотажных изделий встречаются одинаковые значения. Отсортировать исходный массив по возрастанию.

Вариант 3. Для условия задачи варианта 3 главы 11 отсортировать вектор выпуска продукции ткацкими цехами предприятия по убыванию. Добавить в него новые элементы, равные 480, 320 и 520, сохранив упорядоченность.

Вариант 4. Для условия задачи варианта 7 главы 11 найти, сколько раз в векторе количества мушек встречается одинаковое их количество. Вывести на экран отсортированный по возрастанию исходный вектор.

Вариант 5. Для условия задачи варианта 13 главы 11 отсортировать по убыванию значения масс клочков канекарона UHD и RMK. Определить, сколько значений превышают среднюю массу клочка каждого вида волокна.

Вариант 6. Для условия задачи варианта 16 главы 11 отсортировать по возрастанию значения каждого скоростного параметра кипоразрыхлителя. Определить, сколько значений скорости наклонной игольчатой решетки соответствуют оптимальному значению, равному 35,1 м/мин.

Вариант 7. Для условия задачи варианта 7 главы 12 упорядочить по возрастанию артикулы выпускаемых изделий. Определить количество цехов, выпускающих представленные в таблице изделия.

Вариант 8. Для условия задачи варианта 8 главы 12 отсортировать по убыванию значения длин кусков ткани. Определить, сколько кусков имеют максимальную длину.

Вариант 9. Для условия задачи варианта 9 главы 12 упорядочить по возрастанию размер тарифного заработка каждого рабочего. Организовать вывод фамилии работника и его тарифного заработка, причем тарифный заработок должен быть упорядочен.

Вариант 10. Для условия задачи варианта 10 главы 12 отсортировать по алфавиту фамилии работников, имеющих детей. Вывести в одном упорядоченном списке фамилий работников и сведения о наличии детей у каждого из них. Найти, сколько работников имеют более одного ребенка.

ГЛАВА 14. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

При решении инженерно-технических задач довольно часто приходится сталкиваться с зависимостями вида $y_d = f(x)$, где y_d – предельно допустимое или оптимальное значение показателя, а f(x) – функция, выражающая зависимость значения y от x. Например, на складе имеется запас готовой продукции, тонн, объем которой изменяется по закону $y = 6{,}6 \cdot \ln 2x - 9{,}5$, где x – спрос на данную продукцию, тонн. Определить предельное значение объема сбыта, т.е. такое значение x, при котором запас на складе станет равным объему спроса. Для решения данной задачи уравнение будет иметь вид: $x = 6{,}6 \cdot \ln 2x - 9{,}5$.

Для дальнейшего решения уравнение необходимо привести к виду:

$$f(x) = 0, \qquad (14.1)$$

где функция $f(x)$ определена и непрерывна на некотором конечном 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* [10].

Найти корни уравнения (14.1) 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.

Численное решение уравнения (14.1) обычно проводят в два этапа. Сначала необходимо отделить корни, т.е. найти такие достаточно малые области, в каждой из которых заключен только один корень уравнения. Эти области получили название *интервалов изоляции корня*. По своей сути интервал изоляции корня и есть приближенное значение корня уравнения, вычисленное с погрешностью, задаваемой длиной данного интервала. Например, если выделен интервал изоляции корня [0,5; 0,6], то правомерно сказать, что корнем данного уравнения с точностью до 0,1 является значение x=0,5. На втором этапе проводят уточнение отделенных корней с заданной погрешностью ε, используя различные численные методы.

При нахождении интервала изоляции корня средствами языка Паскаль рекомендуется использовать табулирование функции на заданном интервале,

чение значений функции $y=f(x)$ на интервале $[x_{min}; x_{max}]$ при дискретном изменении x, равном dx.

Для уточнения корней существуют многочисленные приближенные методы, из которых рассмотрим *метод простой итерации, метод Ньютона и метод половинного деления.*

При рассмотрении методов будем полагать, 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.

14.1. Метод простой итерации

Уравнение $f(x)=0$ представим в виде

$$x = \varphi(x), \tag{14.2}$$

что всегда можно сделать и притом многими способами. Например, можно выделить из уравнения (14.1) переменную x, остальное перенести в правую часть. Это и будет φ(x). Можно умножить левую и правую часть уравнения на произвольную константу λ, и прибавить к левой и правой частям x, т.е. представить (14.1) в виде

$$x = x + \lambda f(x).$$

При этом $\varphi(x)=x+\lambda f(x)$.

Выберем на отрезке [a; b] произвольную точку x_0 - нулевое приближение. Следующие приближения x_1 и x_2 вычислим по формулам

$$x_1 = \varphi(x_0),\ x_2 = \varphi(x_1),$$

и вообще пусть любое значение x_n получается из x_{n-1} по формуле

$$x_n = \varphi(x_{n-1}). \tag{14.3}$$

Этот процесс последовательного вычисления чисел x_n (n=1, 2, 3, ...) по формуле (14.3) называется *методом итераций* [10]. Геометрическая интерпретация метода представлена на рис. 14.1.

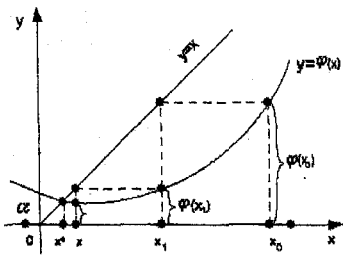


Рис. 14.1. Геометрическая интерпретация метода простой итерации

Если на отрезке [a; b], содержащем корень уравнения (14.2), а также его последовательные приближения x_0, x_1, ..., x_n, ..., вычисляемые по методу итераций, выполнено условие

$$|\varphi'(x)| \le q < 1, \tag{14.4}$$

то процесс итераций сходится, т.е., увеличивая n, можно получить приближенное значение корня, мало отличающееся от истинного значения x*.

Процесс итераций следует продолжать до тех пор, пока для двух последовательных приближений x_{n-1} и x_n не будет выполнено неравенство

$$|x_n - x_{n-1}| \le \frac{1-q}{q}\varepsilon. \tag{14.5}$$

При этом всегда будет выполнено неравенство, обеспечивающее заданную точность ε вычисления корня x*.

$$|x_n - x_{n-1}| \le \varepsilon, \tag{14.6}$$

При практическом нахождении корней по методу итераций следует при переходе от уравнения (14.1) к уравнению (14.2) стремиться представить $\varphi(x)$ так, чтобы производная $\varphi'(x)$ по модулю была меньше единицы. В таком случае корень всегда будет найден, и чем меньше величина q, тем меньшее число итераций для этого потребуется.

Пример 14.1

Решим задачу, изложенную в начале главы. Пусть на складе готовой продукции имеется ее запас, тонн, объем которого изменяется по закону $y = 6{,}6 \cdot \ln 2x - 9{,}5$, где x – спрос на данную продукцию, тыс. тонн. Объемы спроса могут колебаться от 10 до 12 тонн ежедневно. Определить методом итераций с точностью $\varepsilon = 10^{-4}$ такое значение x, при котором запас на складе станет равным объему спроса, и число итераций, необходимое для нахождения корня.

Решение

Для решения данной задачи уравнение будет иметь вид: $x = 6{,}6 \cdot \ln 2x - 9{,}5$. Приведем его к виду (14.1)

$$x - 6{,}6 \cdot \ln 2x + 9{,}5 = 0. \tag{14.7}$$

Это уравнение легко может быть преобразовано к виду $x = \varphi(x)$, где

$$\varphi(x) = 6{,}6 \cdot \ln 2x - 9{,}5. \tag{14.8}$$

Находим $\varphi'(x) = \frac{6{,}6}{x}$. Очевидно, $|\varphi'(x)| = \left|\frac{6{,}6}{x}\right| \le 1$ для всех $10 \le x \le 12$. Поэтому процесс итераций сходится. За начальное приближение можно принять любую точку отрезка [10; 12], например, $x_0 = 11{,}8$.

Алгоритм нахождения корня уравнения (14.8) представляет следующую последовательность действий:

1. Полагаем x = 11,8, ε = 0,0001 и n = 0.
2. Вычисляем следующее приближение x_{n+1} по формуле

$$x_{n+1} = 6{,}6 \cdot \ln 2x - 9{,}5 \tag{14.9}$$

3. Вычисляем разность $\delta = x_{n+1} - x_n$ и увеличиваем величину n на единицу.

4. Проверяем условие $|\delta| < \varepsilon$. Если это условие не выполняется, то возвращаемся к вычислению следующего приближения x_{n+1} по формуле (14.9), т.е. к п.2. Если условие выполняется, то результатом считаем величину x_{n+1} и заканчиваем вычисления. 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. 14.2.

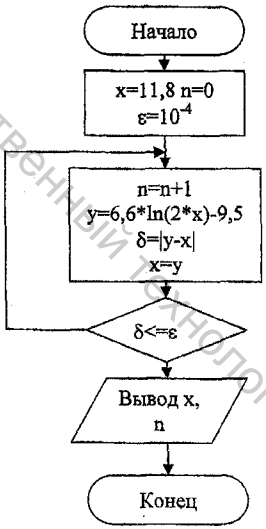


Рис. 14.2. Блок-схема решения уравнения (14.7) методом итераций

Программную реализацию метода итераций представим в двух вариантах:

Вариант 1: с использованием подпрограммы-функции для описания решаемого уравнения, приведенного к виду $x=\varphi(x)$:

```
Program Primer;
Var a,x,y,e,delta:real;
                  n:integer;
{подпрограмма-функция, описывающая уравнение y=6,6*ln(2x)-9,5}
Function F:real;
```

```
Begin
   y:=6.6*ln(2*x)-9.5;
 F:=y;
End;
Begin {Начало раздела операторов основной программы}
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;
{подпрограмма-функция, описывающая уравнение y=6,6*ln(2x)-9,5}
Function F:real;
Begin
   y:=6.6*ln(2*x)-9.5;
 F:=y;
End;
{подпрограмма-процедура, описывающая метод итераций}
Procedure Iter;
 Begin
  n:=0;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:7:4,'  n=',n:2);
End.
```

В обоих случаях получен *результат x=10,7479 n=18.*

Таким образом, при объеме сбыта продукции, равном 10,7479 тонн запасы продукции будут равны спросу на нее.

14.2. Метод Ньютона

Пусть уравнение $f(x)=0$ имеет один корень на отрезке [a; b], причем $f'(x)$ и $f''(x)$ определены, непрерывны и сохраняют постоянные знаки на этом отрезке. Рассмотрение метода Ньютона начнем с его графического представления (рис. 14.3). Возьмем некоторую точку x_0 отрезка [a; b] и проведем в точке $P_0(x_0, f(x_0))$ графика функции касательную к кривой $y=f(x)$ до пересечения с осью Ox. Примем за начальное приближение корня абсциссу x_1 точки пересечения. 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.

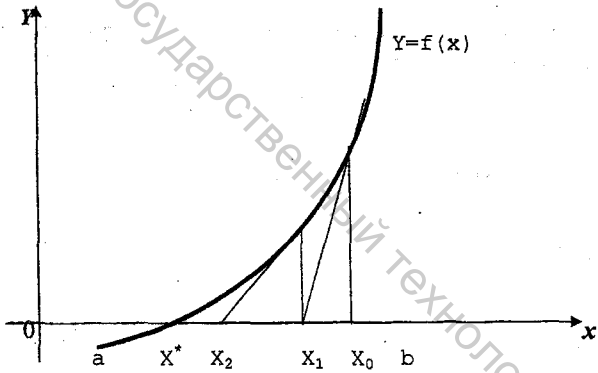


Рис 14.3. Графическое представление метода Ньютона

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

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}, \quad \ldots, \quad x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}.$$

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

$$f(x_0)f''(x_0) > 0. \qquad (14.10)$$

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 (14.10).

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 14.2

По данным [12] производство обуви, млн. пар в год, предприятиями Республики Беларусь за период с 2000 по 2005 г.г. можно описать зависимостью вида $y = 15{,}62 \cdot x^{-0{,}25}$, где x – номер периода (1, 2 и т.д.). Определить, в каком году производство обуви достигнет 9 млн. пар в год, если в 2005 году производство обуви составило 10,1 млн. пар, а прогноз на 2010 год составляет 8,8 млн. пар. Для решения задачи использовать метод Ньютона. Абсолютная погрешность вычисления $\varepsilon=10^{-4}$. Нахождение значений y=f(x) оформить через подпрограмму функции.

Решение

Представим решение задачи в виде уравнения

$$15{,}62 \cdot x^{-0{,}25} = 9.$$

Преобразуем его к виду (14.1):

$$15{,}62 \cdot x^{-0{,}25} - 9 = 0. \qquad (14.11)$$

Определим интервал изоляции корня. По условию задачи производство обуви в 2005 году составило 10,1 млн. пар, а в 2010 году по прогнозам составит 8,8 млн. пар. Следовательно, интервалом изоляции корня является период с 2005 по 2010 год. Если принять 2000 год за период за номером 1, то 2005 году будет соответствовать номер 6, а 2010 – номер 11. Поэтому интервалом изоляции корня данного уравнения будет являться отрезок [6; 11].

Находим $f'(x) = -\frac{3{,}905}{x^{1{,}25}}$. Расчетная формула для уравнения по методу Ньютона имеет вид:

$$x_n = x_{n-1} - \frac{15{,}62 \cdot x^{-0{,}25} - 9}{\left(-3{,}905/x^{1{,}25}\right)},$$

где n=1, 2, 3, …

Начальное приближение x_0 выбираем таким образом, чтобы выполнялось условие $f(x_0)f''(x_0) > 0$. Такой точкой является x=6, так как в ней $f''(x) = \frac{4{,}88}{x^{2{,}25}} > 0$, а значение f(x)>0 для всех x, принадлежащих интервалу [6; 11].

Алгоритм нахождения корня представляет собой следующую последовательность действий:

1. Полагаем $x_0=1$, $\varepsilon=10^{-4}$, $n=0$.
2. Вычисляем следующее приближение:

$$x_{n+1} = x_n - \frac{15{,}62 \cdot x_n^{-0{,}25} - 9}{\left(-3{,}905/x_n^{1{,}25}\right)},$$

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). В противном случае 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 обозначаем через x, x_{n+1} – через y.

Блок-схема реализации метода Ньютона для решения алгебраических и трансцендентных уравнений может быть представлена в следующем виде (рис. 14.4):

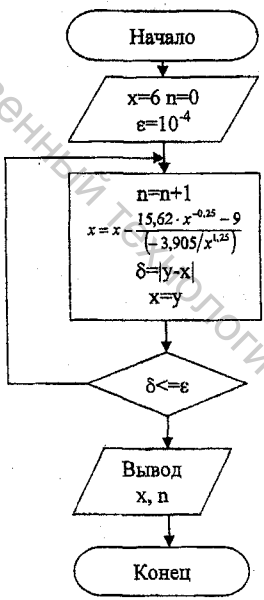


Рис.14.4. Блок-схема решения уравнения (14.11) по методу Ньютона

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;
{подпрограмма-функция для описания уравнения}
 Function F:real;
  Begin
  F:=(15.62*exp(-0.25*ln(x))-9)/(-3.905*exp(1.25*ln(x)));
  End;
{подпрограмма-процедура для реализации метода решения}
Procedure Metod;
Begin
 n:=0;
 Repeat
  n:=n+1;
  y:=x-f;
  delta:=abs(y-x);
  x:=y;
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:7:4, ' n=',n:2);
 Readln;
End.
```

Результат: *x= 9.0485 n=27*

Данному номеру периода соответствует 2008 год, т.е. в 2008 году объем выпуска обуви предприятиями Республики Беларусь составит 9 млн. пар.

14. 3. Метод половинного деления

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 работы, но всегда приводит к результату.

Для нахождения корня уравнения `f(x)=0`, принадлежащего отрезку `[a; b]`, делим отрезок пополам, т.е. выбираем начальное приближение равным x_0=`(a+b)/2`. Если `f(`x_0`)=0`, то x_0 является корнем уравнения. Если `f(`x_0`)`$\neq$`0`, то из двух отрезков `[a; `x_0`]` или `[`x_0`; b]`, выбирают тот, 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 14.3

Методом половинного деления найти корень уравнения

$$f(x)=x-\sqrt{9+x}+x^2-4=0 \qquad (14.12)$$

на отрезке [2; 3] с абсолютной погрешностью $\varepsilon=10^{-4}$.

Решение

Алгоритм нахождения корня уравнения (14.12) представляет следующую последовательность действий:

1. Полагаем a=2, b=3 и ε=0,0001.
2. Вычисляем $f(a)=a-\sqrt{9+a}+a^2-4=0$
3. Вычисляем x=(a+b)/2 и значение функции $f(x)=x-\sqrt{9+x}+x^2-4=0$ в этой точке.
4. Проверяем условие f(x)=0. Если оно истинно, 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(a)*f(x)<0. Если это условие выполняется, то полагаем b=x и переходим к пункту 6. Если условие не выполняется, то полагаем a=x, f(a)=f(x) и переходим к пункту 6.
6. Проверяем условие b-a<=ε. Если оно не выполняется, то возвращаемся к пункту 3. Если условие выполняется, то за результат принимаем значение x и заканчиваем вычисления.

Ниже представлены блок-схема реализации метода половинного деления (рис. 14.5) для решения уравнения (14.12) и программа на языке Паскаль.

```
Program Met_PD;
 Var x,y,z,eps,a,b:real;
     n:integer;
{Подпрограмма-функция для описания исходного уравнения}
 Function F(xx:real):real;
  Begin
  f:=xx-sqrt(9+xx)+sqr(xx)-4
  End;
{Подпрограмма-процедура реализации метода половинного деления}
 Procedure Metod;
 Begin
  y:=f(a); n:=0;
   Repeat
    n:=n+1;
    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;
    Until (b-a)<eps;
  End;
{Основная программа}
  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: $x=2.25775$

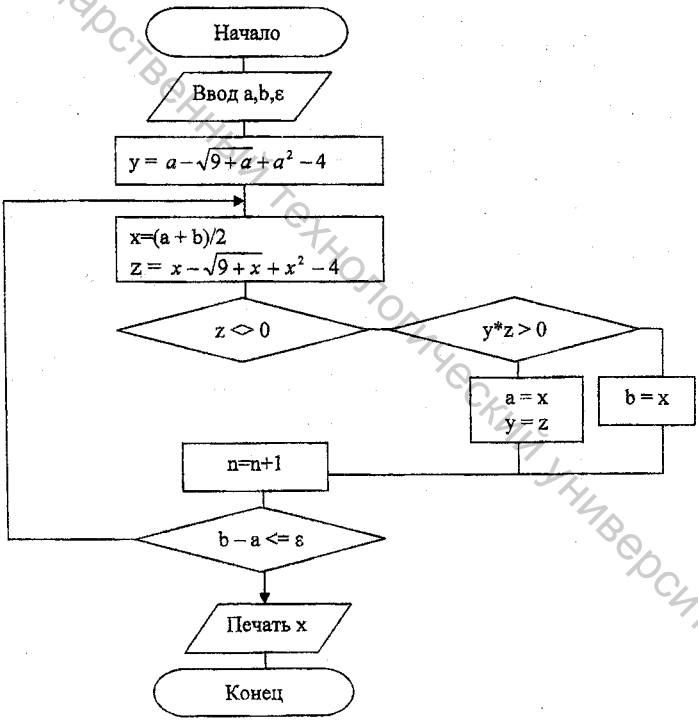


Рис. 14.5. Блок-схема решения уравнения методом половинного деления

14.4. Задания для самостоятельной работы

Выделить интервал изоляции корня уравнения и вычислить корень указанным методом с точностью $\varepsilon = 10^{-4}$ [12, стр. 47].

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 15. ПРИБЛИЖЕННОЕ ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

Довольно часто в научно-технических задачах возникает необходимость вычисления определенного интеграла или значений первообразной функции [10]. Приведем некоторые определения.

1. Функция `F(x)` на данном интервале называется *первообразной функцией* для функции `f(x)`, если на 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 ξ_i `(i=0, 1, …, n-1)`.

Тогда *определенным интегралом* $\int_a^b f(x)dx$ функции `f(x)` на отрезке от a до b называется

$$\lim_{\max \Delta x_i \to 0} \sum_{i=0}^{n-1} f(\xi_i)\Delta x_i .$$

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)` на интервале `[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*.

15.1. 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` (рис.15.1).

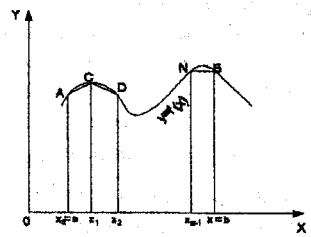


Рис.15.1. Графическая интерпретация определенного интеграла

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, ..., n`. Концы ординат соединим прямолинейными отрезками. Тогда площадь криволинейной трапеции `aABb` приближенно можно считать равной площади фигуры, ограниченной ломаной линией `aACD ... NBb`. Площадь `S` этой фигуры 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

$$\int_a^b f(x)dx\approx\frac{b-a}{2n}\left(y_0+2y_1+\ldots+2y_{n-1}+2y_n\right) \qquad (15.1)$$

При составлении программ целесообразно формулу трапеций (15.1) 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 15.1

Ордината Y развертки верхнего участка мужского воротника определяется по формуле $Y=\int_{V_1}^{V_n}\frac{r^2(r-V)dV}{(2V^2-4rV+r^2)\sqrt{2rV-V^2}}$, где r – радиус, r=1,75 см, V_1=0,05,

V_n=2,25. Определить ординату Y методом трапеций, разбивая отрезок интегрирования на 50 частей.

Решение

Подставим исходные данные в формулу

$$Y=\int_{0,05}^{2,25}\frac{3,0625(1,75-V)dV}{(2V^2-7V+3,0625)\sqrt{3,5V-V^2}}$$

Для вычисления определенного интеграла используются величины

$h_1=\frac{2,25-0,05}{50}=0,04$ — шаг интегрирования;

$V_i=ih_1,\ y_i=\frac{3,0625(1,75-V)}{(2V^2-7V+3,0625)\sqrt{3,5V-V^2}}$, 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 v_i, так как при вычислении очередного элемента суммы требуется лишь одно значение v, которое может быть получено прибавлением шага h к предыдущему значению v.

Далее приводится программа и результат ее выполнения.

```
Program Int_TR;
Var a,b,integr:real;
                n:integer;
{Подпрограмма-функция для описания интегрируемой функции}
Function Fun(z:real):real;
 Begin
   Fun:=(3.0625*(1.75-z))/((2*Sqr(z)-7*z+3.0625)*
Sqrt(3.5*z-Sqr(z)));
 End;
{Подпрограмма-процедура для реализации метода трапеций}
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;
{Основная программа}
 Begin
  Writeln('Введите a,b,n');
  Readln(a,b,n);
```

```
  Integr:=Integral(a,b,n);
  Writeln('Интеграл=',integr:7:4);
End.
```

Результат: *Интеграл = 0.8551*

Таким образом, ордината точки Y развертки мужского воротника равна 0,8551.

15.2.Метод Симпсона

Разобьем отрезок интегрирования [a; b] на 2n равных частей длины $h=\frac{b-a}{2n}$. Пусть точкам разбиения $x_0=a$, $x_1=a+h, \dots$, $x_n=b$ соответствуют значения подынтегральной функции $y_0, y_1, \dots, y_{2n}$, т. е. $y_i=f(x_i)$, $x_i=a+ih$, i= 0, 1, ..., 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

$$\int_a^b f(x)dx\approx\frac{b-a}{6n}(y_0+4y_1+2y_2+\dots+4y_{2n-1}+y_{2n}), \qquad (15.2)$$

известную под названием *формулы Симпсона.*

При составлении программ целесообразно формулу Симпсона (15.2) 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 15.2.

Вычислить интеграл $\int_0^{\pi/2}\sqrt{1-\frac{1}{4}\sin^2 x}dx$ методом Симпсона, разбивая отрезок интегрирования на $2n$=20 частей.

Решение

Для вычисления определенного интеграла имеем

$$h_1 = \frac{\pi/2-0}{2n} = \frac{\pi}{40},\ x_i = ih_1,\ y_i = \sqrt{1-\frac{1}{4}\sin^2 x_i},\ \text{i=0, 1, 2, ..., 20.}$$

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 элемента, 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;
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);
End.
```

Результат: *интеграл*=1.4674

15.3. Задания для самостоятельной работы

Найти значение определенного интеграла. Метод вычисления и количество отрезков задать самостоятельно [12, стр.58].

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	$\int_{\pi/6}^{\pi/3} tg^2x+ctg^2xdx$
4	$\int_0^1 xe^x\sin xdx$	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 16. СИСТЕМА КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAPLE

16.1. 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
– - оператор вычитания
* - умножение
/ - деление
**, ^ - возведение в степень
! – факториал
. – десятичная точка
:= - оператор присваивания
and – логическое И
-> - функциональный оператор
< - менее чем
> - более чем
= - равно
<= - менее чем и равно
>= - более чем и равно
<> - неравно
or – логическое ИЛИ

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);
```

$$3.14$$

16.2. Условные выражения и циклы

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 параметр 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.

16.3. 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).

Пример

```
> 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);
```

$$\frac{\partial^4}{\partial y^2\,\partial x^2}\cos(x)\,y^3 = -6\cos(x)\,y$$

16.4. 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

```
> 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);
```

$$\int_0^{\infty} x\,e^{(-x)}\,dx = 1$$

16.5. 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].

Примеры построения различных видов графиков

1. Построение графика неявно заданной функции `sin(x)/x` на интервале `-15..15` (см. рис. 16.1).

```
> plot(, x=-15..15,color=red, title="график");
```

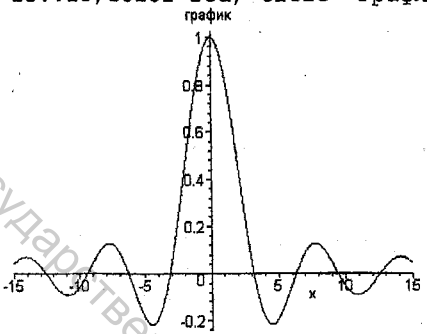


Рис.16.1. График функции `sin(x)/x`

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. 16.2).

```
> fun:=sin(x)^2;
```

$$fun := \sin(x)^2$$

```
> plot(fun, x=-5..5,y=0..0.5,color=black,style=point,
title="График");
```

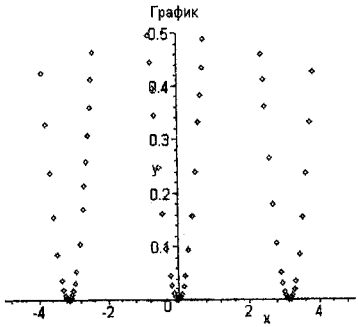


Рис.16.2. График функции `fun=sin^2(x)`

3. Построение графика функции `ln(1+cos(x))`, при x, стремящемся к бесконечности (см. рис. 16.3).

```
>plot(ln(1+cos(x)),x=0..infinity,y=-10..1, color = blue);
```

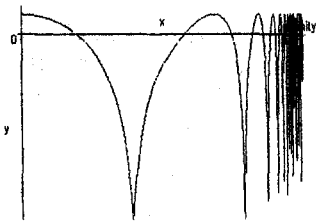


Рис.16.3. График функции `ln(1+cos(x))`

4. Построение графиков трех функций `sin(x)`, `sin(x)/x`, `sin(x^3/100)` линиями трех цветов и трех типов (см. рис. 16.4).

```
> plot([sin(x),sin(x)/x, sin(x^3/100)],
x=-10..10,color=[black,blue,red],style=[line,line,point]);
```

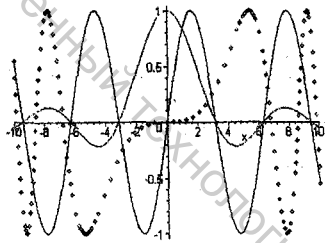


Рис.16.4. График трех функций

5. Построение графика функции, содержащей условное выражение с функцией if (см. рис.16.5).

```
>plot(`if`(sin(x)/x>0,sin(x)/x,-sin(x)/x),x=-10..10,
color=black);
```

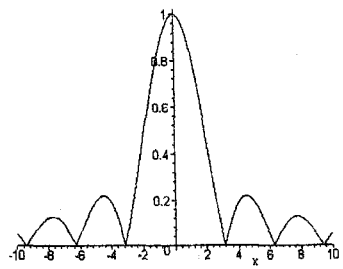


Рис.16.5. График функции, содержащей условное выражение

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);
```

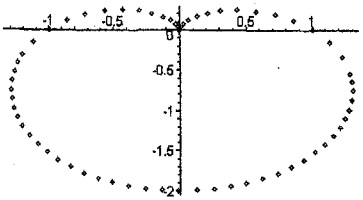


Рис.16.6. График функции в полярной системе координат

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

Построить поверхность h^2 в цилиндрической системе координат (см. рис. 16.7).

```
> plot3d(h^2,a=-Pi..Pi,h=-5..5, coords=cylindrical,
style=patch, color=sin(h));
```

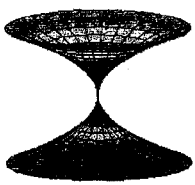


Рис.16.7. Пример трехмерного графика

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.

16.6. 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

```
> 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

```
> 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 [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. 16.8):

```
> plot(3*ln(x+1)-x,x=1..7);
```

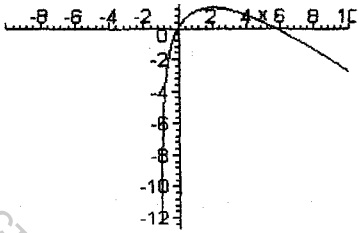


Рис. 16.8. Выделение интервала изоляции корня уравнения

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);
```

$$0., 5.711441084$$

16.7. 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

```
> 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.16.9).

Пример

```
> 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);
```

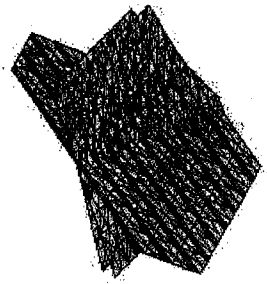


Рис.16.9. Графическое решение системы уравнений

16.8. 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 15, п.15.3), дать графическую интерпретацию решения.

- Решить нелинейное уравнение (см. главу 14, п.14.4). Выделить графически интервал изоляции корня уравнения и вычислить корень с точностью $\varepsilon = 10^{-4}$
- Решить систему уравнений, дать графическую интерпретацию решения:

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 (y) зависит от обрывности нити (x). 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 y относительно x и x относительно y.

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[4]:

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

[4] Здесь и далее в приложениях курсивом обозначен текст, выводимый компьютером.

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.

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.

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

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 покупкȕ 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 значений 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 значений 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. Дьяконов, В. Maple 6 : учебный курс / В. Дьяконов. – Санкт-Петербург : Питер, 2001. – 608 с. : ил.
6. Епанешников, А. Программирование в среде Turbo Pascal 7.0 / А. Епанешников, В. Епанешников. – Москва : Диалог – МИФИ, 2000. – 256 с.
7. Першиков, В. И. Толковый словарь по информатике / В. И. Першиков, В. М. Савинков. – Москва : Финансы и статистика, 1991. – 543 с.
8. Прикладная информатика : учебное пособие / А. Н. Морозевич [и др.] ; под общ. ред. А. Н. Морозевича. – Минск : Вышейшая школа, 2003. – 335 с.
9. Статистический ежегодник Республики Беларусь, 2004 / Министерство статистики и анализа Республики Беларусь. – Минск, 2004. – 660 с.
10. Турчак, Л. И. Основы численных методов / Л. И. Турчак – Москва : Наука, 1987. – 328 с.
11. Интернет ресурс http://www.alfcom.ru
12. Светозарова, Г.И. Практикум по программированию на алгоритмических языках / Г.И. Светозарова, Е.В. Сигитов, А.В.Козловский. – Москва : Наука, 1980. – 320 с.

Учебное издание

Вардомацкая Елена Юрьевна

Окишева Татьяна Николаевна

В двух частях. Часть I.

Учебное пособие

Редактор В.П. Беляев
Технический редактор Е. Л. Афанасьева
Корректор Т. К. Покатович
Компьютерная верстка К.Г. Осипов

Подписано к печати 25.06.2007. Формат 60x84 1/16. Бумага офсетная №1.
Гарнитура «Таймс». Усл.-печ. листов 12,8. Уч.-издат. листов 16,5. Тираж 163 экз. Зак. № 328

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.