Министерство образования и науки
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
«Ивановский государственный энергетический
университет имени В.И. Ленина»

И.Н. СУЛЫНЕНКОВ, Е.Н. КВАША

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
ПО ДИСЦИПЛИНЕ «ИНФОРМАТИКА» ДЛЯ СТУДЕНТОВ ФАКУЛЬТЕТА ЗАОЧНОГО И ВЕЧЕРНЕГО ОБУЧЕНИЯ НАПРАВЛЕНИЯ ПОДГОТОВКИ «ЭЛЕКТРОЭНЕРГЕТИКА И ЭЛЕКТРОТЕХНИКА»

Иваново, 2015

УДК 681.3, 621.311
С 89

Сулыненков И.Н., Кваша Е.Н. Учебно-методическое пособие по дисциплине «Информатика» для студентов факультета заочного и вечернего обучения направления подготовки «Электроэнергетика и электротехника»: Учеб.-метод. пособие/ ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина».– Иваново, 2015.– 100 с.

ISBN

В учебно-методическом пособии приведена программа курса «Информатика». Рассмотрены теоретические вопросы по темам «Системы счисления», «Основы алгоритмизации» и «Язык программирования Паскаль». Приведены методика и примеры решения практических задач. Представлены варианты заданий контрольной работы.

Учебно-методическое пособие предназначено для студентов 1 курса факультета заочного и вечернего обучения направления подготовки «Электроэнергетика и электротехника».

Табл. 12. Ил. 18. Библиогр.: 8 назв.

Печатается по решению редакционно-издательского совета ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»

НАУЧНЫЙ РЕДАКТОР
канд. техн. наук, доцент А.А. Фомичев

РЕЦЕНЗЕНТ
канд. техн. наук, доцент В.М. Лапшин

ISBN

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Информатика» является частью математического и естественно-научного цикла дисциплин по направлению подготовки «Электроэнергетика и электротехника». Она является базовой для освоения предмета «Компьютерные технологии», а также дисциплин, в которых рассматриваются вопросы применения ЭВМ при решении практических задач. Теоретические знания и практические навыки, полученные в ходе изучения дисциплины, потребуются при подготовке расчетно-графических работ, курсовых работ, курсовых проектов и выпускной квалификационной работы.

Учебно-методическое пособие имеет цель ознакомить студента с программой курса, очертить границы изучаемого материала, определить содержание самостоятельной работы, а также необходимую литературу и способ самоконтроля, подготовить студента к выполнению контрольной работы за счет изучения приведенных в указаниях необходимых теоретических разделов курса с иллюстрациями теории на примерах решения практических задач.

Учебно-методическое пособие включает в себя три раздела. В первом разделе приведена программа дисциплины. Отражены разделы, рассматриваемые в курсе «Информатика» при самостоятельной работе и на аудиторных занятиях. Представлены темы дисциплины и рассматриваемые вопросы. По каждой теме приведена рекомендуемая литература. Для самоконтроля представлены вопросы для самопроверки.

Во втором разделе рассмотрены теоретические вопросы и примеры решения практических задач, необходимые для выполнения контрольной работы по курсу. Результатом освоения данного раздела должна стать выполненная контрольная работа. Задания и методические рекомендации по выполнению контрольной работы представлены в третьем разделе учебно-методического пособия.

РАЗДЕЛ ПЕРВЫЙ
Содержание рабочей программы дисциплины «Информатика»

1.1. Основные положения программы дисциплины

Программой дисциплины предусмотрены лекционные занятия, лабораторные занятия и самостоятельная работа студента. Общая трудоемкость освоения дисциплины составляет 4 зачетные единицы или 144 часа. Предусмотрены следующие виды контролей: выполнение контрольной работы, контроль после выполнения лабораторных работ и итоговый контроль в форме экзамена.

Аудиторные занятия включают в себя лекции и лабораторные работы. Основой для проведения аудиторных занятий является внеаудиторная самостоятельная работа студента, которая заключается в проработке теоретических вопросов по учебным материалам дисциплины и Интернет-ресурсам. Названия тем и содержание изучаемых вопросов представлены в разделе 1.3. Студент обязан до начала аудиторных занятий самостоятельно изучить все темы курса и ответить на контрольные вопросы.

Контрольная работа выполняется студентом самостоятельно в период до начала аудиторных занятий и представляется преподавателю на первой лекции. Теория, методика и примеры выполнения работы представлены во втором разделе настоящего учебно-методического пособия. Контрольные задания представлены в третьем разделе. Для разрешения затруднений, возникающих в ходе выполнения контрольной работы, до сессии деканатом ФЗВО назначается одна консультация с преподавателем.

Итоговый контроль студентов проводится по завершении изучения дисциплины в виде экзамена. Форма экзамена – индивидуальное собеседование в сочетании с предварительным тестированием. Допуск к экзамену производится по результатам проверки контрольной работы и отчетам по выполнению лабораторных работ.

1.2. Содержание аудиторных занятий

Аудиторные занятия по дисциплине предусмотрены в первом семестре первого курса.

Теоретические занятия (лекции) по дисциплине проводятся в интерактивном режиме в форме лекции-беседы с возможностью перехода в лекцию-дискуссию или проблемную лекцию. В объем лекционного курса включены следующие основные темы.

1. Операционные системы.
2. Прикладное и системное программное обеспечение.
3. Системы счисления.
4. Основы языка «Турбо Паскаль».
5. Средства компьютерной графики.
6. Проектирование баз данных.

Проведение лабораторных занятий предусмотрено в компьютерных залах отдела компьютерных средств обучения электроэнергетического факультета, и включает в себя следующие работы.

1. Изучение текстового редактора (на примере Microsoft Word, OpenOffice) и/или табличного процессора (на примере Microsoft Excel, OpenOffice).
2. Изучение баз данных (на примере Microsoft Access).
3. Изучение векторного графического редактора (на примере Microsoft Visio, Autodesk AutoCAD).
4. Изучение средств программирования, основанных на языках высокого уровня (на примере Turbo Pascal).

1.3. Самостоятельная работа студента

В данном разделе представлены темы, предназначенные для самостоятельного изучения. Для каждой темы приведен перечень изучаемых вопросов и рекомендуемая литература. По завершению проработки каждой темы следует ответить на вопросы для самопроверки. С их помощью можно оценить полноту изученного материала.

ТЕМА 1. Понятие информации. Общая характеристика процессов сбора, передачи, обработки и накопления информации

Изучаемые вопросы

1. Понятие информации.
2. Информационные процессы и системы.
3. Информационные ресурсы и технологии.
4. Структура информатики и её связь с другими науками.
5. Количество и качество информации.
6. Виды и формы представления информации в информационных системах.
7. Обработка информации.
   7.1. Классификация средств обработки информации.
   7.2. Преобразование аналоговой информации в цифровую информацию.
8. Хранение информации.
   8.1. Классификация запоминающих устройств.
   8.2. Основная память.
   8.3. Внешние запоминающие устройства.
9. Передача информации.
   9.1. Общая схема системы передачи информации.
   9.2. Каналы передачи данных и их характеристика.

Вопросы для самопроверки

1. *Что такое информация?*
2. *Каковы основные свойства информации?*
3. *Что такое информационные технологии?*
4. *Что такое данные?*
5. *Чем отличаются понятия "информация" и "данные"?*
6. *В каких единицах измеряется объем данных?*
7. *Как записываются числа в двоичной и в шестнадцатеричной системах счисления?*
8. *Что такое метод обработки данных?*
9. *Что такое информационный объект?*
10. *Что такое адекватность информации?*

11. *Что такое полнота информации?*
12. *Зачем нужна избыточность информации?*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.11–37], [2, с.15–32].

ТЕМА 2. Технические и программные средства реализации информационных процессов

Изучаемые вопросы

1. Архитектура ЭВМ.
2. Архитектура микропроцессоров.
3. Программное обеспечение ЭВМ.
   3.1. Характеристика основного программного обеспечения: операционных систем, системного программного обеспечения, прикладного программного обеспечения.
   3.2. Характеристика основного прикладного программного обеспечения: систем обработки текстов, табличных процессоров, систем компьютерной графики, инструментальных программных средств для решения прикладных математических задач.

Вопросы для самопроверки

1. *Охарактеризовать основные принципы развития компьютерных систем.*
2. *Каковы отличительные особенности суперкомпьютеров, средних компьютеров и персональных компьютеров, исходя из таких критериев, как производительность, объём памяти, скорость обработки информации, цена и сфера применения?*
3. *В чем состоит назначение основных компонентов персонального компьютера: процессора, винчестера (жёсткого диска), устройств ввода/вывода, различных типов па-*

мяти, сменных носителей информации (дискеты, CD-ROM и т.п.)?

4. *Что такое внешнее устройство?*
5. *Каково назначение процессора (CPU) и каковы выполняемые им операции?*
6. *Чем определяется быстродействие компьютера?*
7. *Что такое тактовая частота процессора? В каких единицах она измеряется?*
8. *Охарактеризуйте основные устройства ввода: клавиатуру, мышь, сканнер, джойстик, микрофон.*
9. *Охарактеризуйте основные устройства вывода: мониторы, принтеры, звуковые колонки.*
10. *В чем состоит различие между типами компьютерной памяти: оперативной (RAM) и постоянной (ROM)?*
11. *Что такое системное и прикладное программное обеспечение? В чем состоит сходство и различие?*
12. *В чем заключаются основные функции операционной системы?*
13. *Что такое графический пользовательский интерфейс? В чем его основные особенности и преимущества?*
14. *Перечислите основные программные приложения и охарактеризуйте сферы их применения - текстовые редакторы, электронные таблицы, базы данных, экономические пакеты (начисление зарплаты и т.п.), презентации, мультимедийные пакеты, издательские системы.*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.42–94], [2, с.394–442].

ТЕМА 3. Алгоритмизация и программирование

Изучаемые вопросы

1. Понятие алгоритма и его свойства.
2. Основные принципы разработки и анализа алгоритмов.
3. Алгоритмические структуры и их использование.

4. Языки программирования. Классификация. Различия. Достоинства и недостатки.
5. Системы программирования.
6. Структурное программирование.
7. Объектно-ориентированное программирование.
8. Основные конструкции языка программирования.
9. Процедуры и функции.
10. Модули.

Вопросы для самопроверки

1. *Перечислите виды алгоритмов и опишите их свойства.*
2. *Что такое структурная блок- и граф-схема алгоритма?*
3. *Какие требования предъявляются к алгоритму?*
4. *Назовите методы проектирования алгоритмов и дайте краткое описание их.*
5. *Что такое модульный метод проектированием алгоритмов и программ? Преимущества модульного проектирования алгоритмов.*
6. *Чем отличаются компилируемые языки программирования от интерпретируемых?.*
7. *На какие уровни по степени детализации делятся языки программирования?*
8. *Перечислите основные языки программирования высокого и низкого уровней.*
9. *Дать определение принципов инкапсуляции, полиморфизма и наследования.*
10. *Что такое класс, свойство и поле в объектно-ориентированном программировании?*
11. *В чем состоит процедурное (императивное) и непроцедурное (декларативное) программирование?*
12. *На какие категории делятся структуры по признаку характера упорядоченности их элементов?*
13. *Какие операции можно проводить над структурами данными?*
14. *Чем отличаются процедуры от функций?*

15. *В чем заключается концепция модульного программирования? Перечислите разновидности модулей.*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.568–608], [2, с.36–42, с.55–63].

ТЕМА 4. Языки программирования высокого уровня

Изучаемые вопросы

1. Основные конструкции языка Turbo Pascal.
   1.1. Алфавит языка.
   1.2. Структура программы на языка.
   1.3. Идентификаторы.
   1.4. Типы данных.
   1.5. Программирование арифметических выражений.
   1.6. Программирование разветвляющихся структур.
2. Программирование циклов.
   2.1. Оператор цикла с параметром «for do».
   2.2. Оператор цикла с предусловием «while do».
   2.3. Оператор цикла с постусловием «repeat until».
3. Организация подпрограмм.
   3.1. Пользовательские функции.
   3.2. Процедуры.

Вопросы для самопроверки

1. *Что такое идентификатор в Pascal? Перечислите основные правила написания идентификаторов.*
2. *Чем отличается тип данных Integer от real?*
3. *Какие служебные слова обрамляют тело основной программы?*
4. *Из каких разделов состоит программа на языке Паскаль?*
5. *Для чего используется оператор присваивания? Формат оператора присваивания.*

6. *Какие операции и соответствующие знаки используются в арифметических выражениях?*
7. *Опишите процедуры ввода и вывода информации.*
8. *Для чего используется оператор «if»?*
9. *Дайте определение цикла и операторов циклического процесса.*
10. *Какие формы записи оператора цикла с параметром в языке Паскаль?*
11. *Какой формат записи оператора цикла с предусловием?*
12. *Какой формат записи оператора цикла с постусловием?*
13. *Назначение пользовательских процедур и функций.*
14. *Опишите структуру процедур в языке Паскаль.*
15. *Опишите структуру функций в языке Паскаль.*

Для самостоятельной работы необходимо изучить раздел 2.3 и соответствующие разделы литературы [3,4].

ТЕМА 5. Программное обеспечение

Изучаемые вопросы

1. Операционные системы.
   1.1. Назначение операционных систем.
   1.2. Основные функции операционных систем.
   1.3. Наиболее распространенные операционные системы. Их характеристика.
2. Системное программное обеспечение.
3. Прикладное программное обеспечение общего назначения.
   3.1. Классификация.
   3.2. Инструментальные программные средства общего назначения.
   3.3. Инструментальные программные средства специального назначения.
   3.4. Программные средства профессионального уровня.
4. Системы обработки текстов. Основы работы в MS Word и текстовом редакторе OpenOffice.

5. Табличные процессоры. Основы работы в MS Excel и табличном процессоре OpenOffice.

Вопросы для самопроверки

1. *Назначение операционной системы. Перечислите наиболее распространенные операционные системы.*
2. *Что такое файловая структура?*
3. *Описать структуру окна программы или папки (область заголовка окна, строка меню, панель инструментов, строка состояния, полосы прокрутки и т.д.).*
4. *Зачем нужны иконки (пиктограммы)?*
5. *Что такое ярлык? Чем он отличается (по виду и по сути) от значка программы?*
6. *Назовите основные текстовые процессоры.*
7. *Как сохранить текущий документ MS Word (OpenOffice) в другом формате (как тестовый файл (*.txt, *.rtf), шаблон или в формате другой версии программы)?*
8. *Как ввести верхние и нижние индексы в MS Word?*
9. *Как перенести фрагмент текста из одного документа MS Word в другой?*
10. *Как изменить тип, размер, начертание (обычный, курсив, полужирный), цвет и дополнительные параметры шрифта в MS Word?*
11. *Как в MS Word задать выравнивание текста в абзаце, межстрочный интервал, отступы, положение абзаца на странице?*
12. *Как найти слово или фрагмент текста в MS Word? Как заменить один фрагмент текста на другой?*
13. *Как в MS Word создать маркированный, нумерованный и другие типы списков? Как отформатировать список?*
14. *Как расположить текст в 2 или в 3 колонки в MS Word?*
15. *Как вставить нумерацию страниц в MS Word?*
16. *Как вставить верхний и нижний колонтитулы в MS Word?*

17. *Как создать таблицу в редакторе MS Word? Как отредактировать в таблице MS Word ячейку, группу ячеек, строку, столбец? Как объединять и разбивать ячейки?*
18. *Как отрегулировать взаимное расположение рисунка и текста?*
19. *Как вставлять и редактировать математические формулы, используя объект Microsoft Equation?*
20. *Что такое электронная таблица?*
21. *Что такое относительные и абсолютные адреса ячеек в MS Excel? Как их использовать?*
22. *Как ввести числа в ячейку электронной таблицы MS Excel? Как ввести текст в ячейку таблицы? Как ввести формулу в ячейку таблицы?*
23. *Как вставлять дополнительные столбцы и строки в электронную таблицу? Как удалять строки и столбцы?*
24. *Как копировать, перемещать и переименовывать рабочие листы в MS Excel?*
25. *Как установить формат отображения содержимого ячейки в MS Excel (числовой, включая количество отображаемых цифр после запятой, формат даты, денежный формат, процентный формат)?*
26. *Как установить и настроить фон и обрамление ячеек?*
27. *Как объединить несколько ячеек в одну в MS Excel?*
28. *Как в MS Excel отсортировать выделенные числовые данные по возрастанию или по убыванию?*
29. *Как в MS Excel построить различные диаграммы по заданным значениям? Как форматировать диаграммы (заголовки, подписи, метки, масштаб, цвета, изменение типа, изменение данных и т.д.)?*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.99–187], [2, с.95–137], [7,8].

ТЕМА 6. Базы данных

Изучаемые вопросы

1. Основные понятия баз данных.
   1.1. Определения.
   1.2. Системы управления базами данных.
   1.3. Компоненты среды СУБД.
2. Архитектура СУБД.
3. Концепции проектирования БД.
4. Модели данных. Реляционная модель.
5. Язык структурированных запросов SQL.
6. Язык запросов по образцу.

Вопросы для самопроверки

1. *Что собой представляет база данных?*
2. *Какие основные преимущества и недостатки у обработки данных на основе технологии баз данных?*
3. *Какой компонент системы баз данных предназначен для создания и ведения баз данных?*
4. *Какой тип имеет поле таблицы MS Access, предназначенное для хранения рисунков?*
5. *Какой тип имеет поле таблицы MS Access, обладающее свойством автоматического наращивания?*
6. *Какой тип имеет поле таблицы MS Access, предназначенное для хранения больших текстовых блоков?*
7. *В каком случае используется поле логического типа в таблице MS Access?*
8. *Дайте определение понятию «первичный ключ отношения».*
9. *Для чего используется внешний ключ отношения (таблицы данных)?*
10. *Какая модель данных использует табличное представление данных?*
11. *Какая модель данных использует для представления данных граф типа «дерево»?*
12. *Какие групповые отношения возможны в иерархической модели данных?*
13. *Какие групповые отношения возможны в сетевой модели данных?*

14. *Что является первичным ключом для отношения СТУДЕНТ, имеющего атрибуты: № зачетки, Фамилия, Группа, Адрес?*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.340–366], [2, с.150–163], [5].

ТЕМА 7. Современные средства компьютерной графики

Изучаемые вопросы

1. Растровая и векторная графики. Отличительные особенности. Достоинства и недостатки.
2. Растровые и векторные редакторы.
3. Объекты рисования и их форматирование.
4. Действия над объектами в графических редакторах.
5. Режимы редактора и способы их переключения.
6. Приемы «простого» рисования из объектов, четкого позиционирования и сопряжения объектов в графических редакторах.
7. Нанесение надписей в графических редакторах.
8. Приемы редактирования (вставка, копирование, удаление и т.д.).
9. Построение изображений с использованием библиотек.

Вопросы для самопроверки

1. *Что такое графический редактор?*
2. *В чем отличие растровых и векторных редакторов?*
3. *Укажите, какие редакторы относятся к растровым графическим редакторам и какие к векторным: Paint, CorelDraw, Adobe Photoshop, Paintbrush, Macromedia Freehand, Adobe Illustrator,Visio, AutoCad.*
4. *Что является минимальным объектом в векторном редакторе? Что является минимальным объектом в растровом редакторе?*

5. *Как создать и настроить рабочий лист в графическом редакторе MS Visio?*
6. *Как изменить заливку объекта в MS Visio?*
7. *Как изменить формат линий объекта в MS Visio?*
8. *Как набрать и редактировать текст в MS Visio?*
9. *Как группировать объекты в MS Visio?*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.398–436], [2, с.138–150], [6].

ТЕМА 8. Локальные и глобальные сети ЭВМ

Изучаемые вопросы

1. Локальные сети.
   - 1.1. Аппаратные средства локальных сетей.
   - 1.2. Организация передачи данных в локальных сетях.
   - 1.3. Протоколы передачи данных.
2. Глобальная сеть интернет.
   - 2.1. Принципы организации.
   - 2.2. Коммуникационное оборудование.
   - 2.3. Протоколы передачи данных.
   - 2.4. Информационные ресурсы и сервисы Интернета.

Вопросы для самопроверки

1. *Что такое локальные(LAN) и глобальные (WAN) сети?*
2. *Что такое Internet?*
3. *Назовите основные протоколы передачи данных.*
4. *Что такое протокол IP? Его назначение.*
5. *Протоколы TCP и UDP. Назначение. Достоинства и недостатки.*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.195–214], [2, с.451–488].

ТЕМА 9. Основы защиты информации

Изучаемые вопросы

1. Понятие о компьютерной безопасности. Основные угрозы.
2. Брандмауэры. Назначение. Принципы функционирования.
3. Защита информации. Принцип достаточности защиты.
4. Шифрование информации.
5. Основные понятия криптографии.
6. Понятие об электронной подписи.
7. Понятие об электронных сертификатах.
8. Компьютерные вирусы. Средства антивирусной защиты.
9. Защита информации в интернете.

Вопросы для самопроверки

1. *Зачем необходимо резервирование информации?*
2. *В чем заключается защита от несанкционированного доступа?*
3. *Что такое пароль? Как осуществляется защита паролей от несанкционированного доступа?*
4. *Что такое компьютерный вирус? Как он может проникнуть в компьютер?*
5. *Какие существуют способы предотвращения воздействия вирусов на компьютер?*
6. *Чем отличаются коммерческое, свободно распространяемое (free-ware) и частично-распространяемое (share-ware) программное обеспечение?*
7. *Как правильно включать и выключать ПК? Чем грозит неправильное выключение ПК?*
8. *Что необходимо предпринимать при сбое и при "зависании" компьютера? Как использовать диспетчер задач, вызываемый комбинацией клавиш Ctrl+Alt+Del?*

Для самостоятельной работы необходимо изучить соответствующие разделы литературы [1, с.215–224].

РАЗДЕЛ ВТОРОЙ
Теория, методики и примеры по практической части контрольной работы

2.1. Системы счисления

2.1.1. Базовые понятия о системах счисления

Система счисления представляет собой способ кодирования числовой информации, т.е. записи чисел с помощью некоторого алфавита, символы которого называют цифрами. Под числом следует понимать меру описания количества чего-либо.

Представление чисел арабскими цифрами – самая распространённая система счисления. Она называется десятичной, так как использует десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9. Повсеместное использование десятичной системы счисления объясняется историческими предпосылками и физиологическими особенностями человека. Издревле при счете использовались пальцы рук. У человека на каждой руке по пять пальцев, на двух - десять. Было бы на руке шесть пальцев, считали бы мы не десятками, а дюжинами.

Количество цифр, используемых в системе счисления, называется её «основанием». В десятичной системе основание равно десяти.

Десятичные числа, значения которых превышают девять, записываются двумя и более цифрами. Например, число сто двадцать девять записывается тремя цифрами – 129. При этом появляются разряды, значения каждого из которых изменяется в диапазоне от 0 до 9. Любое десятичное число можно представить в виде суммы произведений значений разрядов на основание системы счисления в степени номера разряда. Нумерация разрядов производится справа налево, начиная с нуля. Например, число 126 есть

$$126=1*10^2+2*10^1+6*10^0=1*100+2*10+6*1,$$

или, говорят, одна сотня, два десятка, шесть единиц. Таким образом, в десятичном числе разряды отвечают за единицы, десятки, сотни и т.д., при этом младший (наименее значимый) разряд стоит справа. Например, в числе 1356 четыре значащих разряда: тысяч, сотен, десятков и единиц, – значения которых равны соответственно 1, 3, 5 и 6. Наименее значимый разряд единиц стоит справа, т.е. запись числа производится слева направо от старшего разряда к младшему. Можно дополнить число нулевыми разрядами слева, отвечающими за десятки тысяч, сотни тысяч и миллионы, т.е. 0001356. Дополнительной информации они не несут, поэтому при записи числа их отбрасывают.

Десятичная система счисления является позиционной, т.е. от места цифр в числе зависит его значение. Например, числа 12 и 21, записанные с использованием двух одинаковых цифр 1 и 2, но по-разному расставленных в числах, имеют разные значения.

Десятичная система счисления является привычной для нас. Однако ее использование в электронной технике затруднительно. Полупроводниковые транзисторы, являющиеся базовыми элементами электронных схем, надежно работают только с двумя уровнями сигнала (устойчивыми физическими состояниями): есть напряжение на входе, нет напряжения на входе. Их принято называть единица и ноль. Таким образом, для записи разряда числа можно использовать только две цифры, а значит, система счисления электронной техники является двоичной. Двоичные числа записываются с использованием арабских цифр 1 и 0. Они же используются и в других системах счисления. Отсюда возможны ошибки в интерпретации чисел. Например, число 10 в двоичной системе счисления равно трем, а в десятичной – десяти. Поэтому при записи числа будем указывать нижним индексом справа от числа основание системы счисления, в которой оно приведено. Для представленного примера, числа в двоичной и десятичной системах счисления должны быть записаны: 10_2 и 10_{10}. В языках программирования десятичные числа обычно пишут без обозначения системы счисления, а двоичные числа дополняют строчной латинской буквой b на конце, например, 1001010b.

Принцип формирования двоичных чисел аналогичен принципу формированию десятичным числам. Разряды двоичного числа принято называть битами. Младший (наименее значимый) бит двоичного числа расположен справа, старший – слева. При достижении битом максимального значения, равного 1, и увеличении его на единицу, значение этого бита обнуляется, а в следующий по старшинству (расположенный слева) бит добавляется единица. На рис. 2.1 показаны схемы изменения чисел при достижении одним из разрядов максимального значения и его увеличении на единицу. Для понимания схем они приведены как для двоичных, так и для десятичных чисел. При проведении операции сложения можно пользоваться стандартной схемой сложения «столбиком». Такой же стандартный подход можно применить к операциям вычитания, умножения и деления.

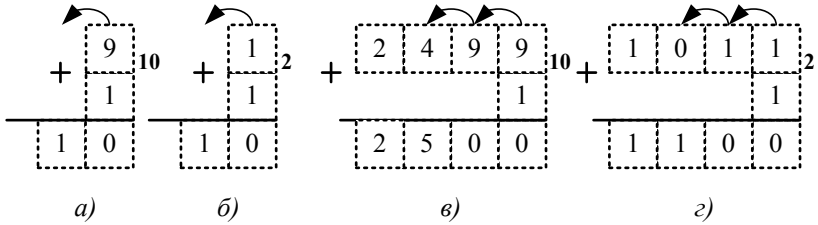


Рис. 2.1. Схемы изменения чисел при их увеличении на единицу:
а – десятичного одноразрядного числа; *б* – двоичного одноразрядного числа; *в* – десятичного четырехразрядного числа; *г* – двоичного четырехразрядного числа

Компьютерам очень просто оперировать двоичными числами. Однако люди не привыкли работать с большим количеством цифр. Например, чтобы представить обыкновенное число сто двадцать шесть тысяч пятьсот тридцать шесть $126\,536_{10}$ в двоичном формате потребуется 17 двоичных цифр (11110111001001000_2). Поэтому возникла потребность создания систем счисления, которые, с одной стороны, были бы удобны представлением числа меньшим количество разрядов, а с другой – легко трансформировались бы в двоичное число. Самый простой способ реализации такого подхода – разбить двоичное

число на несколько разрядов и заменить их цифрами. Если разбить двоичное число по три разряда потребуется 2^3=8 цифр, а если по четыре – 2^4=16. Поэтому были придуманы восьмеричная и шестнадцатеричная системы счислений. В восьмеричной системе счисления для записи разрядов чисел используется восемь арабских цифр: 0, 1, 2, 3, 4, 5, 6 и 7. В шестнадцатеричной системе счисления требуется шестнадцать цифр, что больше десяти. Поэтому в дополнении к десяти арабским цифрам добавляют шесть прописных латинских букв. Отсюда шестнадцать цифр системы счисления: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E и F. Разбивать двоичное число по два разряда не оправдано, так как запись сократится всего в два раза, а при разбиении по пять разрядов потребуется 32 цифры, что значительно затруднит понимание числа.

Информация в ячейки оперативной и постоянной памяти, а также в регистры процессора записываются не по одному биту, а байтами. Один байт равен восьми битам. Даже если информация может быть представлена всего одним битом, в памяти она будет отображаться целым байтом, т.е. в этом случае семь старших бит байта будут равны нулю, а значащим будет только младший бит. При записи восьми бит требуется дробное количество восьмеричных чисел (8/3) и целое количество шестнадцатеричных (8/4). Отсюда один байт можно легко записать двумя шестнадцатеричными числами, что нельзя сказать о восьмеричных числах. Поэтому шестнадцатеричная система счисления нашла более широкое применения, даже не смотря на то, что при записи чисел используются не привычные нам буквенные цифры. В языках программирования для обозначения того, что число записано в шестнадцатеричной системе счисления, его дополняют строчной латинской буквой h на конце, например, 537h или 0D0Ah.

Таким образом, существует четыре наиболее распространенные системы счисления. Возникает потребность знания правил перевода чисел из одной системы счисления в другую. Этому вопросу посвящен раздел 2.1.2 учебно-методического пособия и первый раздел практической части контрольной работы.

2.1.2. Правила перевода чисел между системами счисления

Преобразование десятичных чисел в двоичные числа

Перевод чисел из десятичной системы счисления в двоичную необходим для преобразования понятных пользователю десятичных чисел к двоичным, с которыми работает компьютер. Для перевода используют так называемый «алгоритм замещения», состоящий из следующей последовательности действий:

1. Делим десятичное число на **2** нацело. Остаток от деления (единица или ноль) запоминаем.
2. Если частное от деления на предыдущем шаге не равно **0**, принимаем его за новое делимое и повторяем процедуру, описанную в шаге 1.
3. Алгоритм продолжается до тех пор, пока в результате выполнения шагов 1 и 2 не получится частное, равное нулю, и остаток, равный единице. После этого остатки от деления (**0** или **1**) записываются в разряды двоичного числа в направлении от ***первого*** остатка к ***последнему справа налево***.

Схема перевода десятичного числа в двоичное число приведена на рис. 2.2.

Преобразование двоичных чисел в десятичные числа

Задача перевода чисел из двоичной системы счисления в десятичную возникает при обратном преобразовании вычисленных компьютером значений в более понятные пользователю десятичные числа. Алгоритм перевода двоичных чисел в десятичные числа достаточно прост. Для этого необходимо представить число в виде суммы произведений степеней числа два (2^n, где n – степень, соответствующая порядковому номеру бита начиная с правого бита заканчивая левым, нумерация начинается с нуля), на соответствующие цифры (ноль или единица) в разрядах двоичного числа. Схема преобразования двоичных чисел в десятичные представлена на рис. 2.3.

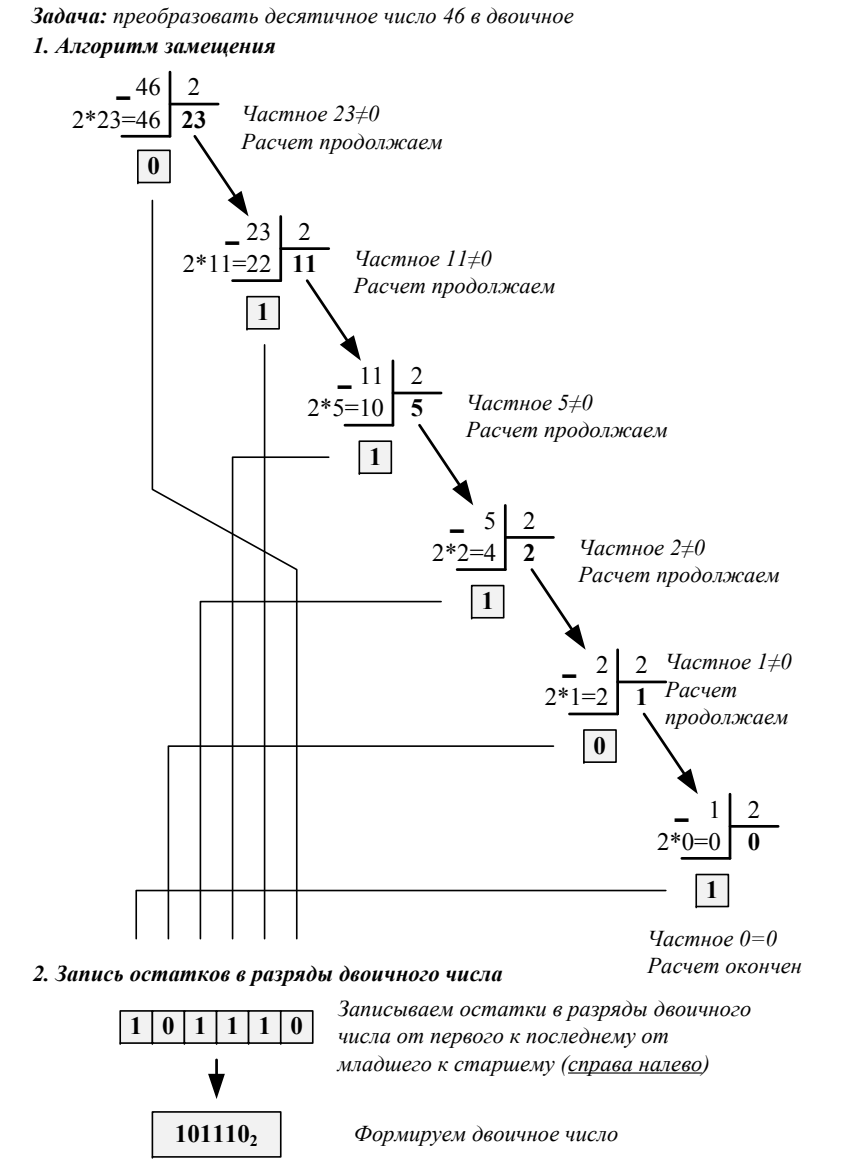


Рис.2.2. Схема преобразования десятичного числа в двоичное число

Задача: *преобразовать двоичное число 1111101000 в десятичное*

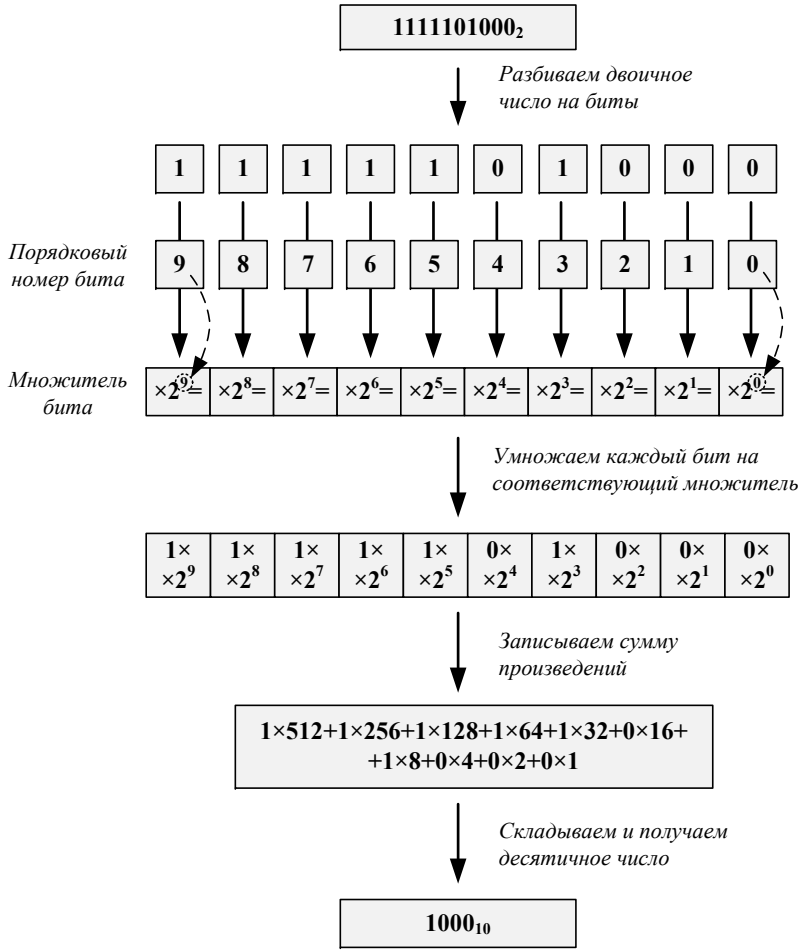


Рис.2.3. Схема преобразования двоичного числа в десятичное число

Преобразование десятичных чисел в шестнадцатеричные числа

Для перевода чисел из десятичной системы счисления в шестнадцатеричную используют тот же «алгоритм замещения», что и при переводе из десятичной системы счисления в двоичную, только в качестве делителя используют число **16** – основание шестнадцатеричной системы счисления.

1. Делим десятичное число на **16** нацело. Остаток от деления запоминаем. Если он больше 9, то вместо цифр используются строчные латинские буквы A, B, C, D, E, F, причем 10=A, 11=B, 12=C, 13=D, 14=E, 15=F.
2. Если частное после шага 1 не равно **0**, принимаем его за новое делимое и повторяем процедуру, описанную в шаге 1.
3. Алгоритм продолжается до тех пор, пока в результате выполнения шагов 1 и 2 не получится частное, равное **нулю**. Остатки записываются в разряды шестнадцатеричного числа в направлении от ***первого*** к ***последнему справа налево***.

Схема преобразования десятичного числа в шестнадцатеричное число представлена на рис. 2.4.

Преобразование шестнадцатеричных чисел в десятичные числа

Для перевода шестнадцатеричного числа в десятичное необходимо это число представить в виде суммы произведений степеней основания шестнадцатеричной системы счисления 16 (16^n, где n – степень, соответствующая порядковому номеру разряда шестнадцатеричного числа начиная с правого, заканчивая левым, нумерация начинается с нуля) на соответствующие цифры в разрядах шестнадцатеричного числа. Строчные латинские буквы A, B, C, D, E, F следует предварительно заменить цифрами A=10, B=11, C=12, D=13, E=14, F=15.

Схема преобразования шестнадцатеричного числа в десятичное число представлена на рис. 2.5.

Задача: *преобразовать десятичное число 32 766 в шестнадцатеричное*

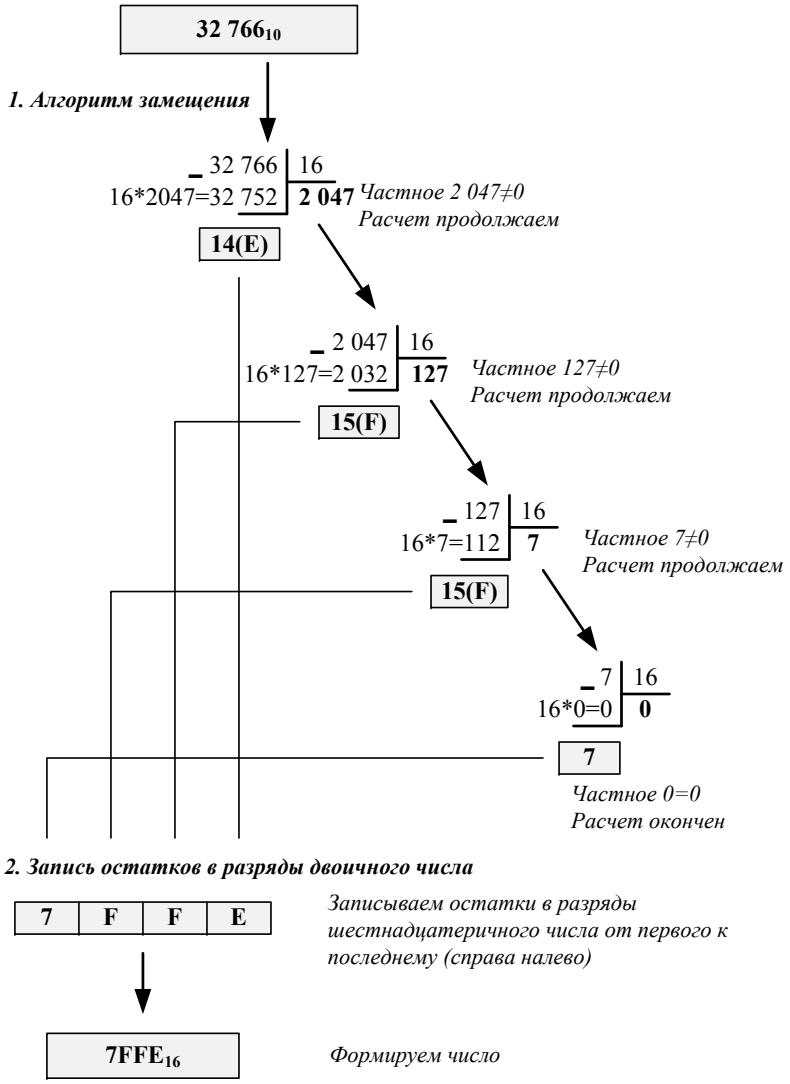


Рис.2.4. Схема преобразования десятичного числа в шестнадцатеричное число

Задача: *преобразовать шестнадцатеричное число F45ED23A в десятичное*

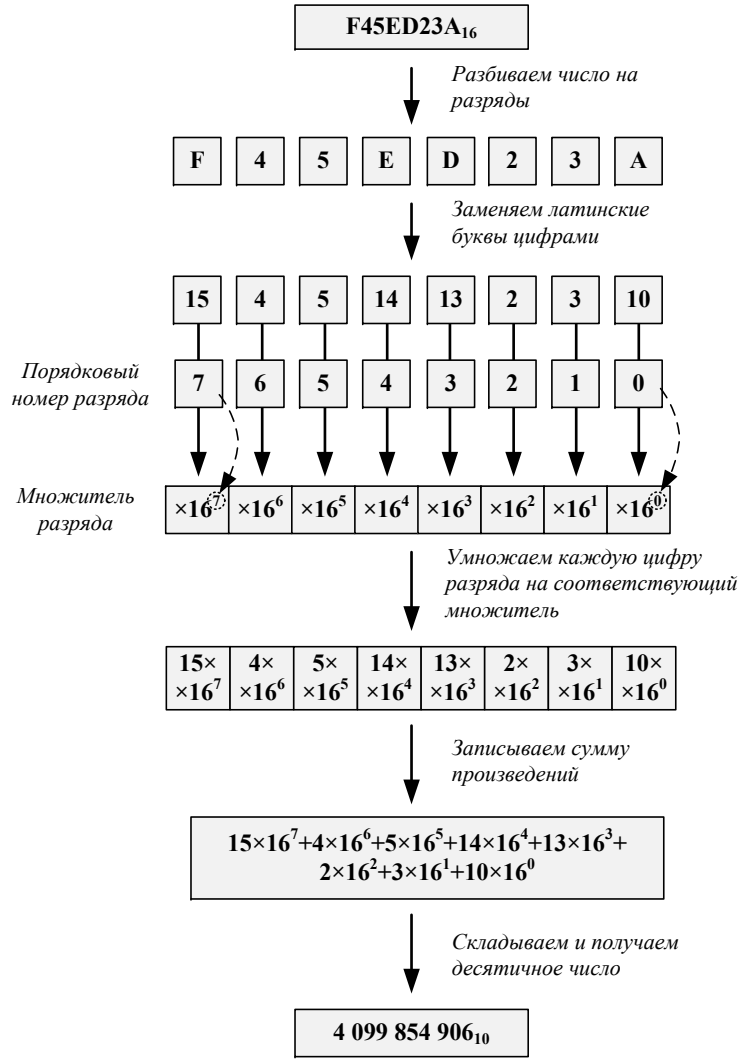


Рис.2.5. Схема преобразования шестнадцатеричного числа в десятичное число

Преобразование десятичных чисел в восьмеричные числа

Для перевода чисел из десятичной системы счисления в восьмеричную используют тот же «алгоритм замещения», что и при переводе из десятичной системы счисления в двоичную, только в качестве делителя используют число **8** – основание восьмеричной системы счисления.

1. Делим десятичное число на **8** нацело. Частное запоминаем для следующего шага, а остаток записываем как разряд восьмеричного числа.
2. Если частное не равно **0**, принимаем его за новое делимое и повторяем процедуру, описанную в шаге 1.
3. Алгоритм продолжается до тех пор, пока в результате выполнения шагов 1 и 2 не получится частное, равное **0**. Каждый новый остаток записывается в разряды восьмеричного числа в направлении от ***первого*** к ***последнему справа налево***.

Схема преобразования десятичных чисел в восьмеричные представлена на рис. 2.6.

Преобразование восьмеричных чисел в десятичные числа

Для перевода восьмеричного числа в десятичное необходимо это число представить в виде суммы произведений степеней основания восьмеричной системы счисления 8 (8^n, где n – степень, соответствующая порядковому номеру разряда восьмеричного числа начиная с правого, заканчивая левым, нумерация начинается с нуля) на соответствующие цифры в разрядах восьмеричного числа.

Схема преобразования восьмеричных чисел в десятичные представлена на рис. 2.7.

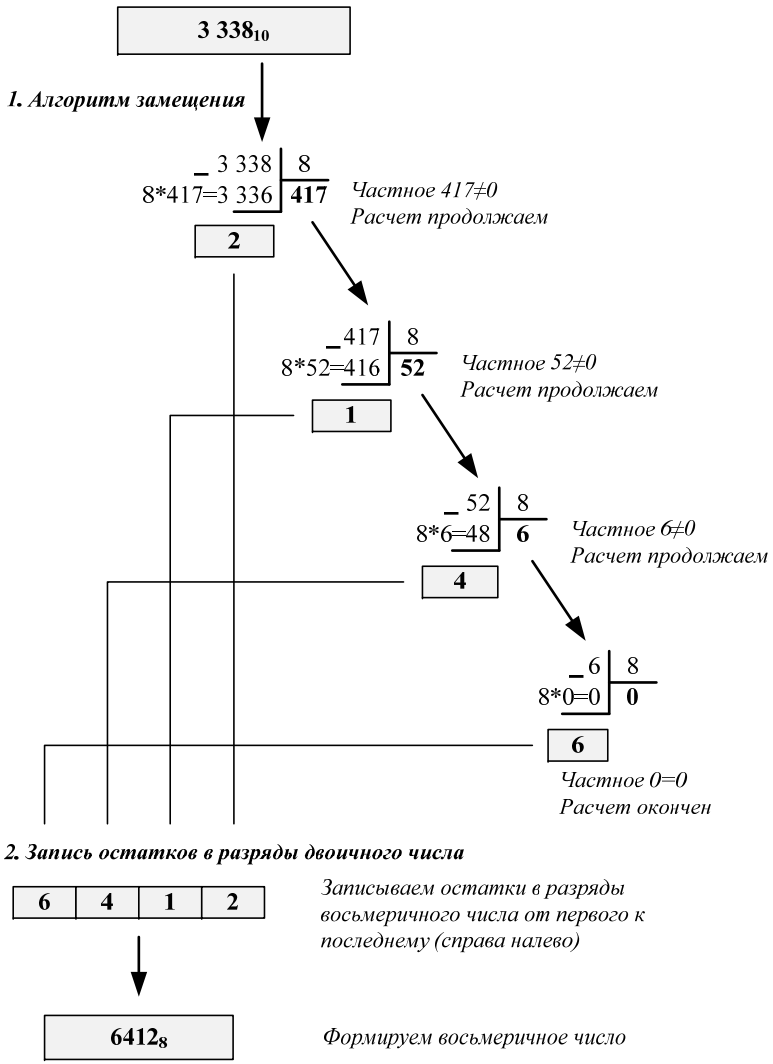


Рис.2.6. Схема преобразования десятичного числа в восьмеричное число

Задача: *преобразовать восьмеричное число 2357 в десятичное*

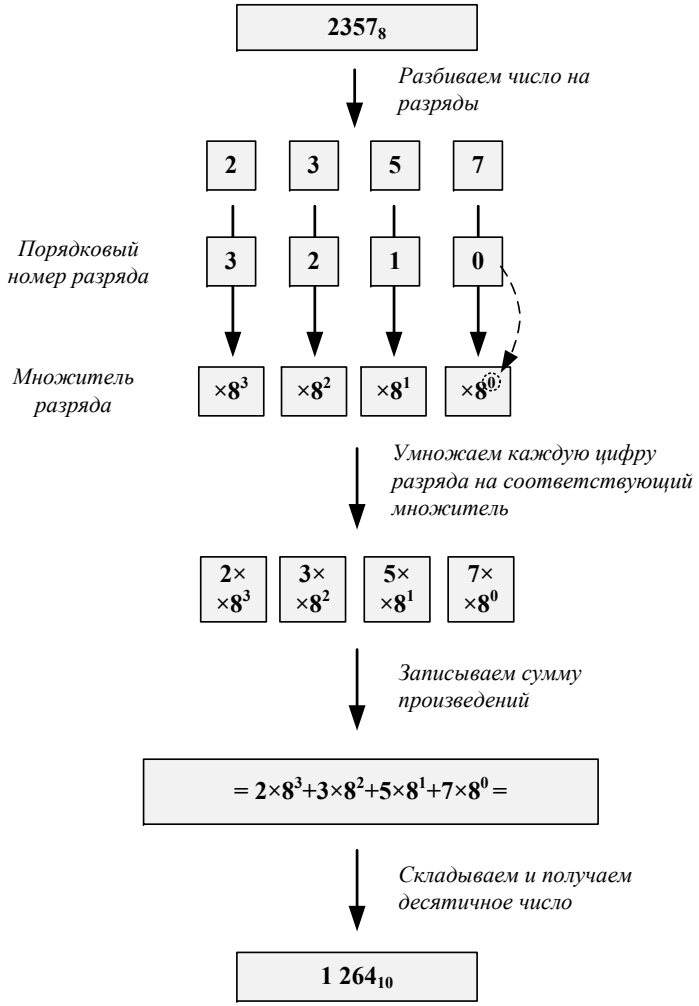


Рис.2.7. Схема преобразования восьмеричного числа в десятичное число

Преобразование двоичных чисел в восьмеричные и шестнадцатеричные числа

Для перевода двоичного числа в восьмеричную систему счисления следует разбить это двоичное число на триады (по три бита) начиная с младшего (правого) бита. Если старшая (левая) триада не заполнена до конца, следует дописать в ее старшие разряды нули. После этого необходимо заменить двоичные триады на числа, равные им в восьмеричной системе в соответствии с таблицей перевода (табл. 2.1).

Таблица 2.1. **Таблица перевода двоичных чисел в восьмеричные**

Двоичные триады	000_2	001_2	010_2	011_2	100_2	101_2	110_2	111_2
Восьмеричные эквиваленты	0_8	1_8	2_8	3_8	4_8	5_8	6_8	7_8

Аналогично поступаем при переводе чисел из двоичной системы счисления в шестнадцатеричную систему счисления. Только разбиение двоичного числа производим не на триады, а на тетрады (по четыре бита). Заменять двоичные тетрады необходимо на числа, равные им в шестнадцатеричной системе счисления в соответствии с таблицей перевода (табл. 2.2).

Схемы преобразования двоичных чисел в восьмеричные и шестнадцатеричные числа представлены на рис. 2.8 и рис. 2.9 соответственно.

Преобразование шестнадцатеричных чисел в двоичные и восьмеричные числа

Алгоритм перевода чисел из шестнадцатеричной системы счисления в двоичную крайне прост (рис. 2.10). Необходимо только заменить каждую цифру шестнадцатеричного числа ее эквивалентом в двоичной системе счисления. Удобно при этом воспользоваться таблицей соответствия (табл. 2.2). Каждое шестнадцатеричное число следует заменять двоичным, представленным

четырьмя битами (т.е. старшие – левые нули каждой тетрады не отбрасываются до получения шестнадцатеричного числа).

Таблица 2.2. **Таблица перевода двоичных чисел в шестнадцатеричные**

Тетрады двоичной системы	Шестнадца-теричные эквиваленты		Тетрады двоичной системы	Шестнадца-теричные эквиваленты
0000_2	0_{16}		1000_2	8_{16}
0001_2	1_{16}		1001_2	9_{16}
0010_2	2_{16}		1010_2	A_{16}
0011_2	3_{16}		1011_2	B_{16}
0100_2	4_{16}		1100_2	C_{16}
0101_2	5_{16}		1101_2	D_{16}
0110_2	6_{16}		1110_2	E_{16}
0111_2	7_{16}		1111_2	F_{16}

При переводе чисел из шестнадцатеричной в восьмеричную систему счисления вначале шестнадцатеричное число переводят в двоичное число. Затем разбивают его на триады начиная с младшего бита и заменяют триады соответствующими им эквивалентами в восьмеричной системе по схеме, приведенной выше.

Преобразование восьмеричных чисел в двоичные и шестнадцатеричные числа

Алгоритм перевода восьмеричных чисел в двоичные состоит в простой замене чисел одной системы на равные им числа другой системы счисления (в случае положительных чисел). На начальном этапе удобно и полезно воспользоваться таблицей соответствия (табл. 2.1).

Для перевода чисел из восьмеричной системы в шестнадцатеричную в начале переводят восьмеричное число в двоичное, а затем уже в шестнадцатеричное по алгоритму, описанному ранее. Пример перевода восьмеричного числа в двоичное и шестнадцатеричное числа представлен на рис. 2.11.

Задача: *преобразовать двоичное число 11100110110111100101011001001 в восьмеричное*

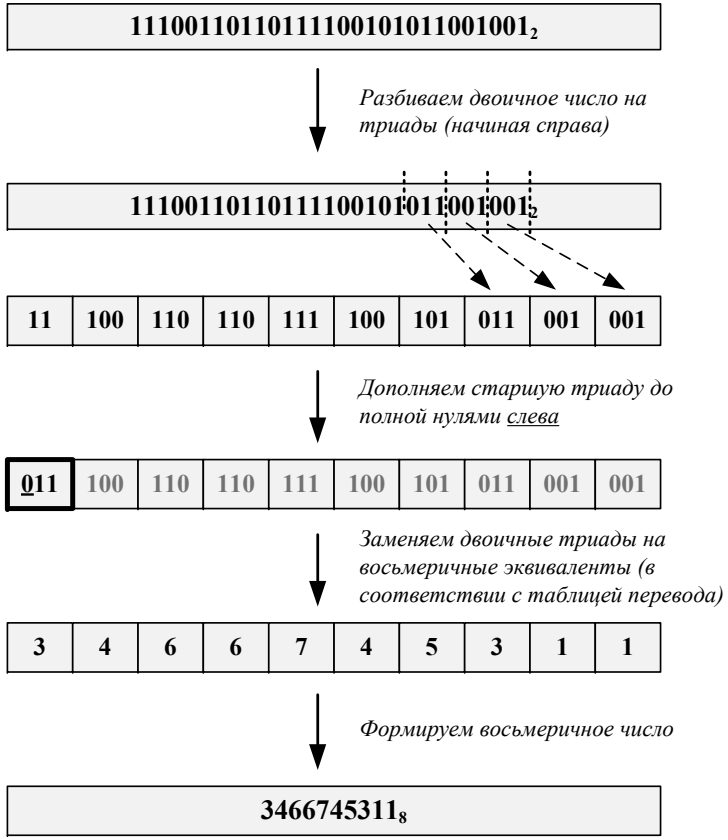


Рис.2.8. Схема преобразования двоичного числа в восьмеричное число

Задача: *преобразовать двоичное число 11100110110111100101011001001 в шестнадцатеричное*

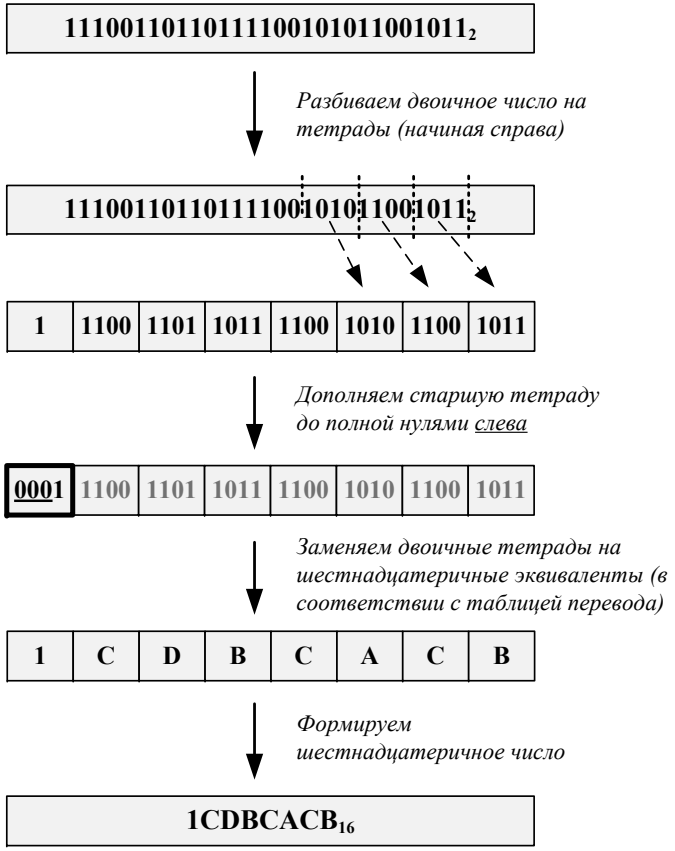


Рис.2.9. Схема преобразования двоичного числа в шестнадцатеричное число

Задача: *преобразовать шестнадцатеричное число 2FC1AB10 в двоичное и восьмеричное*

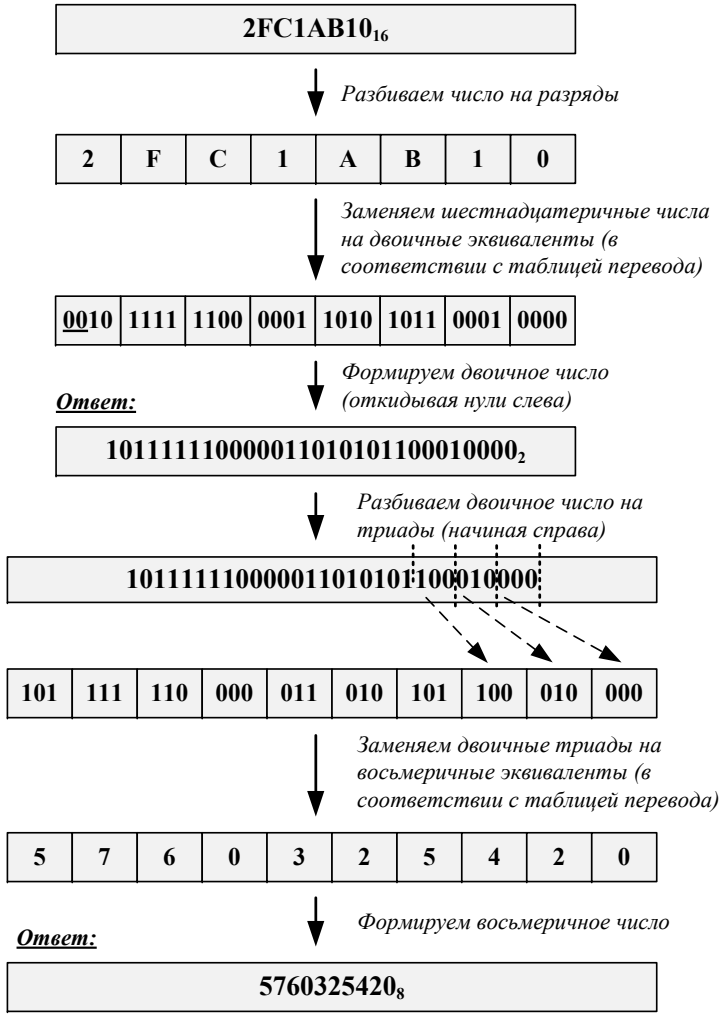


Рис.2.10. Схема преобразования шестнадцатеричного числа в двоичное и восьмеричное числа

Задача: *преобразовать восьмеричное число* ***24715302*** *в двоичное и шестнадцатеричное*

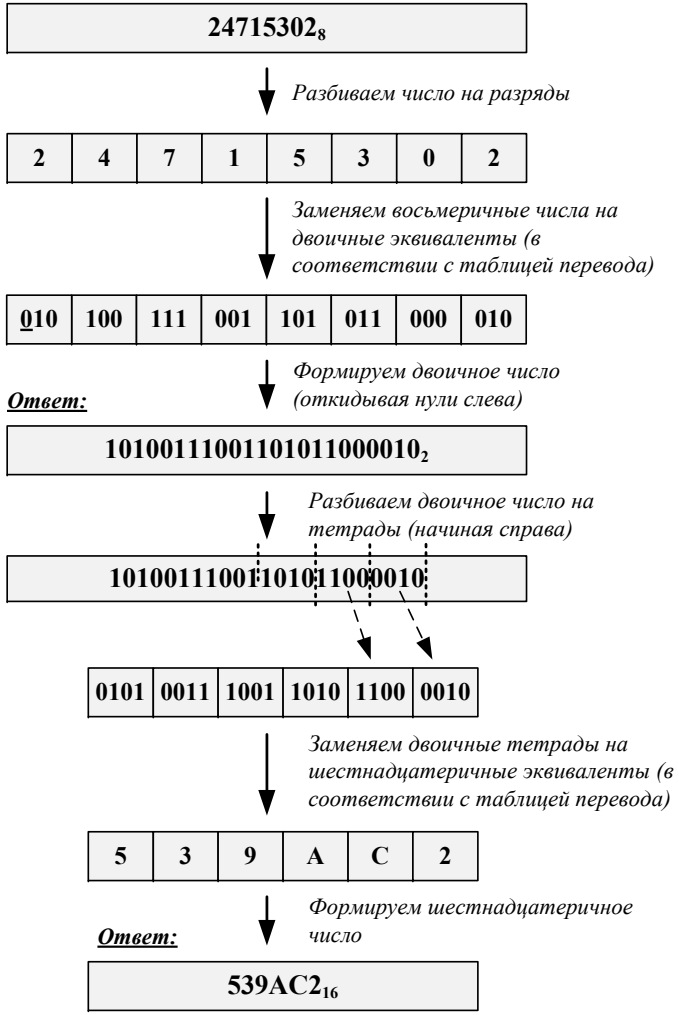


Рис.2.11. Схема преобразования восьмеричного числа в двоичное и шестнадцатеричное числа

2.2. Основы алгоритмизации

Алгоритм – это предписание некоторому исполнителю выполнить конечную последовательность действий, приводящую к некоторому результату. В программировании алгоритм является фундаментом программы, а основным исполнителем – компьютер. На стадии тестирования алгоритма исполнителем может быть сам программист.

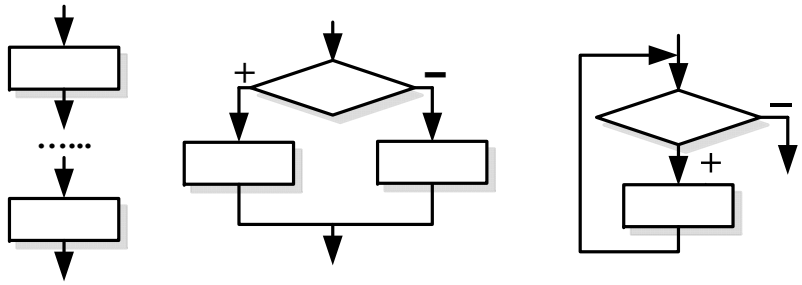
Алгоритм может быть записан с помощью блок-схемы, текстовым предписанием, с помощью рисунков, таблично или на специальном алгоритмическом языке. Наиболее популярны блок-схемы и предписания.

Для изображения алгоритмов будем использовать блок-схемы. Их преимущество заключается в наглядности алгоритма. Блок-схемы формируют из типовых блоков, представленных на рис. 2.12.

Рис.2.12. Типовые блоки алгоритмов

В теории алгоритмов доказано, что любой, сколь угодно сложный алгоритм может быть составлен из трех основных ал-

горитмических структур: линейной, ветвления и цикла, показанных на рис. 2.13, *а*; 2.13, *б* и 2.13, *в* соответственно.



а) *б)* *в)*

Рис.2.13. Алгоритмические структуры:
а – линейная; *б* – ветвление; *в* – цикл

Линейная структура предполагает последовательное выполнение действий, без их повторения или пропуска некоторых действий. Обычно программисты стремятся к тому, чтобы алгоритм имел линейную структуру.

Структура «ветвление» предполагает выполнение одной из двух групп действий в зависимости от выполнения условия в блоке ветвления. На рис. 2.13, *б* знаком «+» показано выполнение условия, а знаком «—» – его невыполнение. Часто используется неполная команда ветвления, когда один из блоков действия отсутствует.

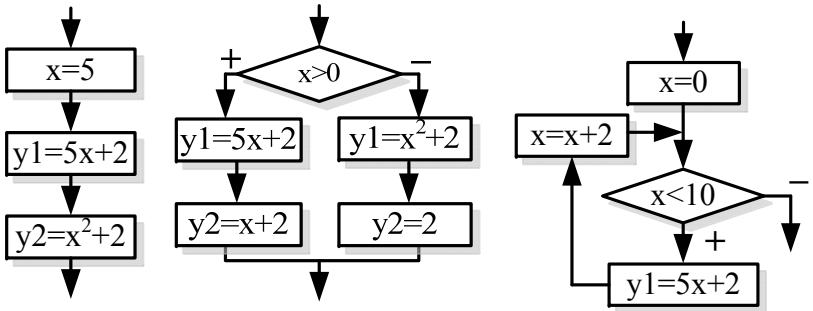
Структура «цикл» имеет несколько разновидностей. На рис. 2.13, *в* показан цикл типа «пока» с предусловием. Действия внутри этого цикла повторяются, пока выполняется условие в блоке ветвления, причем сначала проверяется условие, а затем выполняется действие.

В языках программирования имеются команды, реализующие показанные выше структуры.

При разработке блок-схемы допускается делать любые записи внутри блоков, однако эти записи должны содержать достаточно информации для выполнения очередных действий.

На рис. 2.14 представлены примеры использования алгоритмических структур.

1. Линейный блок алгоритма, производящий расчет значений двух функций при значении аргумента x=5 (рис. 2.14, *а*).
2. Блок ветвления, в котором производится расчет значения функции в зависимости от значения аргумента x (рис. 2.14, *б*). Если x>0, выполняется левый линейный блок, в противном случае – правый.
3. Блок цикла, в котором пять раз производится вычисление значения функции при значении аргумента x в диапазоне [0;10) с шагом 2 (рис. 2.14, *в*).

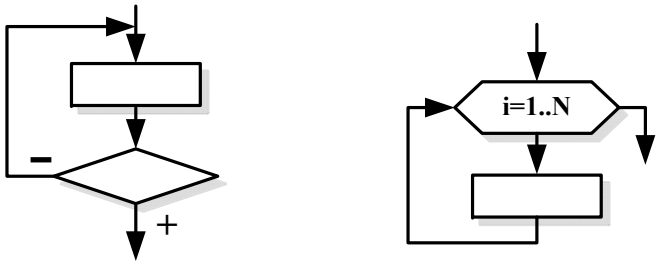


а) *б)* *в)*

Рис.2.14. Примеры использования алгоритмических структур

Кроме цикла типа «пока», достаточно часто используются другие типы циклов. В цикле с постусловием (рис. 2.15, *а*) проверка условия выхода из цикла выполняется после очередного действия. При этом действие будет выполнено минимум один раз. Цикл «для значений в диапазоне» (2.15, *б*) является модификацией цикла «пока» для ситуации, когда заранее известно количество повторений некоторых действий. Запись в блоке заголовка цикла на рис. 2.15, *б* показывает пример описания заголовка цикла, в котором действия повторяются столько раз, сколько целых значений приобретает параметр цикла i от своего

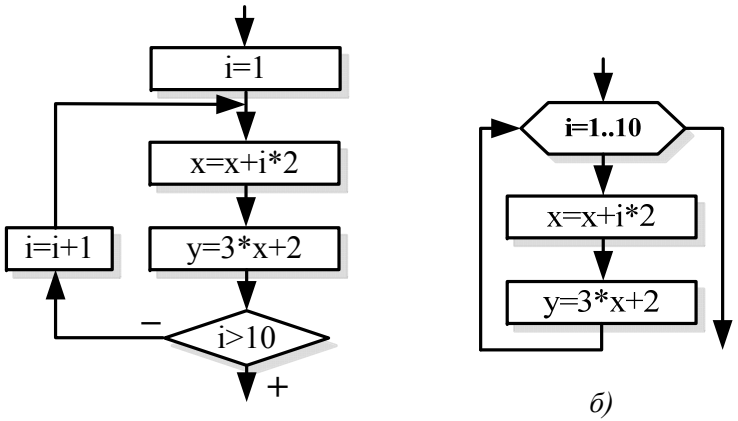
начального значения 1 до конечного N с шагом 1. Обычно шаг не указывается, если он равен 1.



а) *б)*

Рис. 2.15. Разновидности циклов:
а – цикл «пока» с постусловием; *б* – цикл «для значений в диапазоне»

На рис. 2.16 представлены примеры использования различных видов циклов. В цикле рис. 2.16, *а* производится вычисление значения функции *y* при *x*=1..10 с шагом 1. Аналогичная задача реализована в цикле рис. 2.16, *б*.



а) *б)*

Рис. 2.16. Примеры использования циклов:
а – цикл «пока» с постусловием; *б* – цикл «для значений в диапазоне»

Рассмотрим пример создания алгоритма. Задача – вычислить значение функции *f(x)* для заданного пользователем аргумента *x*. Значение функции определяется по формуле

$$f(x)=\begin{cases}3x+1, \text{если } x\le 0\\ x^2+1, \text{если } x>0\end{cases}. \quad (1)$$

Вычисление продолжать до момента, пока пользователь не захочет завершить программу.

Вначале необходимо предоставить пользователю возможность ввести значение аргумента *x*, по которому будем вычислять значение функции. Для этого создадим блок ввода (рис 2.17, *1*).

Затем в зависимости от введенного значения необходимо произвести вычисление функции. Если x>0, вычисление производим по второму выражению формулы 1, в противном случае – по первому. Воспользовавшись блоком «ветвление» (блок *2* рис. 2.17), получим блок алгоритма, реализующего описанный принцип.

После произведенных вычислений следует вывести результат на экран (блок *3* рис. 2.17).

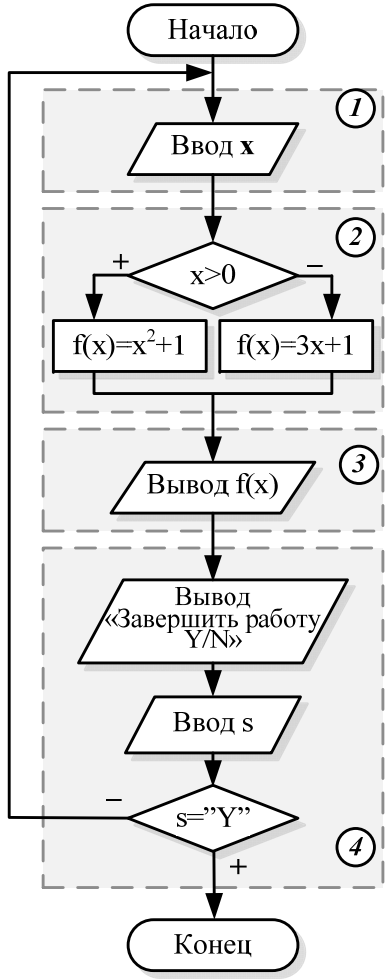


Рис. 2.17. Алгоритм решения задачи: *1* – блок ввода исходных данных; *2* – блок вычисления функции; *3* – блок вывода вычисленного значения; *4* – блок повторения программы

Одним из условий задачи является многократное вычисление функции до момента, пока пользователь не захочет завершить программу. Для этого необходимо создать блок цикла. Так как вычисление функции должно быть произведено хотя бы один раз, используем цикл «пока» с постусловием (рис. 2.17, *4*). В блоке повторения программы выводим пользователю запрос на окончание программы и даем пользователю ввести значение переменной *s*, которое может иметь значение «Y» - продолжить или «N» - завершить. В зависимости от введенного значения либо возвращаемся к началу программу, либо завершаем ее.

Добавив к представленным блокам блоки начала и конца программы, получаем алгоритм решения задачи (рис. 2.17).

2.3. Основы языка программирования «Паскаль»

2.3.1. Краткое введение

Решение многих современных задач сопряжено с реализацией сложных алгоритмов и большим объемом вычислений. Например, расчет режима разветвленной электрической сети потребует многих миллионов операций сложения, вычитания, умножения и деления. Применение устного счета для решения таких задач крайне затруднительно и трудоемко. Появление современных электронно-вычислительных машин (компьютеров) позволило в миллионы раз сократить время, необходимое для вычислений. Однако компьютер сам по себе интеллектом не обладает и является лишь средством решения задачи. Для работы компьютеров необходимо создавать программы, по которым компьютер будет производить расчет.

Программа – это детальное и законченное описание последовательности действий средствами языка программирования. Исполнителем программы является компьютер. Для того чтобы компьютер выполнил программу, она должна быть представлена в машинном коде – последовательности двоичных чисел, понимаемых процессором. Написать программу в машинных кодах

вручную достаточно сложно. Поэтому сегодня практически все программы создаются с помощью языков программирования, которые своими синтаксисом и семантикой приближены к естественному человеческому языку. Это снижает трудоемкость программирования и облегчает ее понимание.

Текст программы, записанный с помощью языка программирования, должен быть преобразован в машинный код. Эта операция выполняется с помощью специальной служебной программы, называемой транслятором. Подобный подход можно сравнить с руководством подчиненным через переводчика, переводящего ваши слова дословно. Схема общения такая: вы даете команду на русском языке (языке программирования), переводчик (транслятор) переводит ваш текст на язык собеседника (в машинный код), собеседник (процессор) выполняет вашу команду.

Языки программирования схожи с обычными человеческими языками: русским, английским и другими. Существуют буквы, из которых составляют слова. Слова должны быть в правильном порядке расставлены в предложениях, должны быть правильно расставлены знаки препинания и т.д. Иначе ваш собеседник вас не поймет или поймет неправильно. Тоже самое можно сказать о компьютерной программе. Незнание языка программирования можно сравнить с общением с иностранцем, разговаривающем на неизвестном вам языке. При этом факт разговора вы видите, но смысла сказанного при этом не понимаете.

Наиболее популярными и универсальными языками программирования являются Basic (Бейсик); Pascal (Паскаль); C++ (Си плюс-плюс); Java (Ява).

По способу разработки программ можно выделить два подхода.

1. Процедурное программирование – это программирование, при котором выполнение команд программы определяется их последовательностью, командами перехода, цикла или обращениями к процедурам.
2. Объектно-ориентированное программирование – программирование, при котором формируются программные объекты, имеющие набор свойств, обладающие набором

методов и способные реагировать на события, возникающие как во внешней среде, так и в самом объекте (нажатие мыши, срабатывание таймера, превышение числовой границы и т.д.).

Объектно-ориентированное программирование (ООП) не исключает технологию процедурного программирования. Наоборот, она является основой ООП. Поэтому изучение принципов программирования правильно начать с процедурно-ориентированных языков, одним из которых является Турбо Паскаль. Ранее он применялся для написания программ для операционной системы MS DOS, работающей без графической оболочки, т.е. с командной строки. Со временем он преобразовался в объектно-ориентированный язык программирования Object Pascal, а впоследствии язык программирования Delphi. С их помощью можно создавать программы, работающие под управлением операционной системы Microsoft Windows.

2.3.2. Алфавит и словарь языка программирования Паскаль

Изучение любого языка принято начинать с его алфавита, т.е. набора знаков, состоящих из букв, цифр и специальных символов. Алфавит Паскаля составляют:

1) прописные и строчные буквы латинского алфавита: «A», «B», «C»…«Y», «Z», «a», «b», «c»,…«y», «z»;
2) десятичные цифры: «0», «1», «2»,…,«9»;
3) специальные символы: «+», «–», «*», «/», «>», «<», «=», «;», «#», «‘», «,», «.», «:», «{», «}», «[», «]», «(», «)», «^»;
4) комбинации специальных символов, которые нельзя разделять пробелами, если они используются как знаки операций: «:=», «..», «<>», «<=», «>=», «{}».

Неделимые последовательности знаков алфавита образуют слова, отделенные друг от друга разделителями. Ими (разделителями) могут быть пробел, комментарий или символ конца строки. Словарь Паскаля можно разделить на три группы слов: зарезервированные слова, стандартные идентификаторы и идентификаторы пользователя.

Зарезервированные слова имеют фиксированное написание и навсегда определенный смысл. Они не могут изменяться программистом и их нельзя использовать в качестве имен для обозначения величин. Зарезервированные слова будем обозначать полужирным курсивом. Перечень основных зарезервированных слов: ***absolute, and, array, begin, case, const, div, goto, do, downto, else, end, file, for, function, if, interrupt, interface, label, library, mod, not, or, of, object, procedure, program, repeat, string, then, to, type, until, uses, var, while, with, xor***.

Идентификатор – имя (identification – установление соответствия объекта некоторому набору символов). Идентификатор можно сравнить с фамилией, именем и отчеством человека, по которым можно обратиться к конкретному человеку. Для обозначения определенных разработчиками языка функций, констант и т.д. служат стандартные идентификаторы, например, *Sqr*, *Sqrt*. В этом примере *Sqr(x)* вызывает функцию, которая возводит в квадрат число *x*, а *Sqrt(x)* – корень квадратный из заданного числа. Пользователь может переопределить любой стандартный идентификатор, но чаще всего это приводит к ошибкам, поэтому на практике их используют без изменения. Идентификаторы пользователя – это те имена, которые дает сам программист.

При записи программ нужно соблюдать общие правила написания идентификаторов.

1. Идентификатор начинается только с буквы (исключение составляют специальные идентификаторы меток).
2. Идентификатор может состоять из букв, цифр и знака подчеркивания.
3. Максимальная длина – 127 символов.
4. При написании идентификаторов можно использовать прописные и строчные буквы.
5. Между двумя идентификаторами должен стоять хотя бы один пробел.

Любой идентификатор пользователя, используемый в исполняемых операторах, должен быть предварительно описан в разделе описаний программы. Описать идентификатор – это

значит указать тип связанного с ним объекта программы (константы или переменной). Этот процесс можно сравнить со знакомством ваших родителей с вашими друзьями. Вы показываете на человека, называете его имя и даете соответствующее описание (друг, коллега по работе и т.д.). Аналогично при описании идентификаторов вы «знакомите» с ними вашу программу. Стандартные идентификаторы, как старых друзей, предварительно описывать не следует.

Идентификаторы должны быть уникальными, т.е. не иметь повторений в названии, иначе возможно неоднозначное обращение. Приведем пример из жизни. Если вы говорите о товарище с именем Андрей, а друзей с именем Андрей у вас несколько, для понимания о ком вы говорите, у вас спросят его фамилию.

2.3.3. Структура программы на языке Паскаль

В языке программирования Паскаль программа состоит из заголовка, раздела описаний и исполняемой части. Структура программы:

program *<имя_программы>*;	Заголовок
uses *<подключаемые_модули>*; ***label*** *<перечень_меток>*; ***const*** *<константы>=<их_значения>*; ***type*** *<описание_новых_типов_переменных>*; ***var*** *<переменные>:<их_тип>*; *<Определение_процедур>* *<Определение_функций>*	Раздел описаний
begin *<Тело_программы>* ***end.***	Исполняемая часть

В приведенной структуре программы вместо тестовых описаний на русском языке вида *<имя_программы>* необходимо

задать пользовательские идентификаторы по требованиям, приведенным выше.

Блок ***«program»*** – блок описания заголовка программы. Пример:

program *my_program*;

В Паскале этот блок имеет декоративное значение и может отсутствовать. Следует отметить, что Турбо Паскаль не чувствителен к регистру, т.е. игнорируется различие в высоте букв (заглавные или строчные), если только это не связано с текстовыми константами. Например, ***program***, ***Program***, ***PROGRAM*** или ***PrOgRaM*** для компилятора есть одно и то же.

Блок ***«uses»*** – перечень дополнительно подключаемых вспомогательных готовых модулей (подпрограмм), собранных в библиотеки (модули). Например, процедуры рисования точек, линий, окружностей на экране содержатся в модуле *graph*. Пример:

uses *crt, graph*;

Если подключать дополнительные библиотеки не нужно, блок отсутствует. Большинство важнейших ключевых системных библиотек подключаются автоматически (по умолчанию).

Блок ***«label»*** – блок описания меток, который содержит их имена, перечисленные через запятую. Метки используются для организации переходов в программе. Пример:

label *l1, l2*;

Если метки не нужны, блок отсутствует.

Блок ***«const»*** – блок описания простых и типизированных констант с указанием их значений. Пример:

```
const
    chislo_pi=3.14;
    dney_v_nedele=7;
```

Блок может отсутствовать, если константы в программе не предусмотрены.

Блок ***«type»*** – блок описания дополнительных типов данных, используемых в программе.

type
moi_tip=***array*** [1..5] ***of*** *byte*;

Блок может отсутствовать, если новые типы не вводятся.

Блок ***«var»*** – блок описания переменных с указанием их типа. Пример:

var
k,n: *integer*;
c: *char*;

Блок может встречаться в программе несколько раз для организации глобальных и локальных переменных (т.е. в основной программе и при описании процедур и функций).

Определение *процедур и функций* – специально оформленные вспомогательные алгоритмы в виде подпрограмм. Вопрос создания процедур и функций не входит в базовый курс, поэтому в данном учебно-методическом пособии не рассматривается.

Блок ***«begin end.»*** – служебные слова, обрамляющие тело основной программы, где находятся исполняемые операторы. ***«begin»*** начинает исполняемую часть программы, а ***«end.»*** (точка в конце обязательна) ее завершает. Теоретически в минимально возможном наборе программа может состоять только из пустого тела ***«begin end.»***

2.3.4. Константы

Константа является разновидностью идентификатора, предварительно заданное значение которой в теле программы изменить нельзя, т.е. она обладает свойством «только для чтения». Объявление констант осуществляется в блоке «***const***». Запись начинается с зарезервированного слова ***const***, сообщающего компилятору о начале описания констант. Точка с запятой после ***const*** не ставится. Далее производится объявление констант по схеме <*имя_константы*>=<*ее_значения*>. Каждая новая константа отделяется от предыдущей точкой с запятой. Имя константы может быть любым (задается программистом), но оно должно подчиняться правилам описания идентификаторов. Значение константы указывается через символ «=». Значение может быть целым, вещественным или шестнадцатеричным числом, логическим, символом, строкой символов и рядом других.

Константы, являющиеся целыми числами, записываются со знаком или без него и могут иметь значение от -2 147 483 648 до +2 147 483 647. Следует учесть, что, если целочисленная константа выходит за указанные границы, компилятор дает сообщение об ошибке. Такие константы должны записываться с десятичной точкой, т.е. определяться как вещественные числа.

Константы, являющиеся вещественными числами, записываются со знаком или без него с использованием десятичной точки (например, 3.14) и/или экспоненциальной части (например, 3E-1). Экспоненциальная часть начинается символом *e* или *E,* за которым могут следовать знаки «+» или «-» и целое число. Символ *e (E)* означает десятичный порядок и имеет смысл «умножить на 10 в степени». Например, 3.14E5 есть 3.14 умножить на 10 в степени 5, а -17E-2 есть минус 17 умножить на 10 в степени минус 2.

Константы, являющиеся шестнадцатертными числами, состоят из шестнадцатеричных цифр, которым предшествует знак доллара $. Диапазон шестнадцатеричных чисел - от $00000000 до $FFFFFFFF.

Логическая константа – это либо слово *false* (ложь), либо слово *true* (истина).

Символьная константа – это любой символ, заключенный в апострофы. Например, ’W’, ’1’, ’в’.

Строковая константа – любая последовательность символов (кроме символа *CR* – возврат каретки), заключенная в апострофы. Если в строке нужно указать сам символ апострофа, он удваивается. Пример строковой константы: ’*Это константа*’. Строка символов может быть пустой, т.е. не иметь никаких символов в обрамляющих ее апострофах.

Пример записи блока констант:

const
c1=4E2;
moya_const=’*Константа*’;

Такое объявление констант фактически дает директиву компилятору считать, что в любом месте тела программы при указании идентификатора *c1* считать его равным числу 4E2, а при указании *moya_const* – строке '*Константа*'.

2.3.5. Переменные

Переменные являются разновидностью идентификаторов, объявляемые в блоке «***var***». В отличие от констант, переменные могут изменять свое значение во время выполнения программы. Описание переменных начинается с зарезервированного слова ***var***, сообщающего компилятору о начале блока объявления переменных. Точка с запятой после ***var*** не ставится. Далее производится объявление всех используемых в программе переменных по схеме *<имя_переменной>:<ее_тип>*. Имя переменной может быть любым (задается программистом), но подчиняющимся правилу описания идентификаторов. Переменные, имеющие один тип, обычно группируют и перечисляют через запятую по схеме *<переменная1>,<переменная2>,…:<тип>*. Однако это не обязательно. После имен переменных ставится символ «:» и указывается их тип. Тип может быть стандартным или пользовательским. Стандартные типы данных описаны в разделе 2.3.6. Вопрос создания пользовательских типов выходит за границы базового курса и не рассматривается. Каждое новое объявление переменной или группы переменных одного типа отделяется от предыдущего точкой с запятой. Порядок объявления переменных в блоке ***var*** не имеет значение.

Пример записи блока переменных:

```
var
    per_1, per_2: byte;
    s: string;
    b: byte;
```

Этот блок можно описать следующим образом: объявляются идентификаторы *per_1, per_2, b*, которые в программе могут принимать значения в диапазоне [0;255], и строковая переменная *s*, состоящая из набора символов.

2.3.6. Стандартные типы данных

Любые данные в Паскале, т.е. константы, переменные, значения функций или выражения, характеризуются своими типами. Тип определяет множество допустимых значений, которые может иметь тот или иной объект, а также множество допустимых операций, которые применимы к нему. Кроме того, тип определяет также и формат внутреннего представления данных в памяти компьютера. Типы данных делятся на три основных вида: простые, структурированные и строки.

1. Простые типы данных включают в себя вещественные, целые, логический и символьный типы, а также тип-диапазон.

Вещественные типы

Вещественный тип предназначен для объявления переменных, значения которых будут дробными числами. Запись таких чисел осуществляется до некоторого знака после запятой, зависящей от внутреннего формата. В табл. 2.3 приводится названия вещественных типов, длина их представления в байтах, количество цифр, которыми будет записано число, и диапазон возможных значений.

Таблица 2.3. **Вещественные типы данных и их параметры**

Название	Длина, байт	Количество цифр	Диапазон значений
real	6	11…12	-39,0…38,0
double	8	15…16	-324,0…308,0
extended	10	19…20	-4 951,0…4 932,0
comp	8	19…20	$-2\cdot10^{63}…2\cdot10^{63}$

Особое положение занимает тип *comp*, который трактуется как вещественное число без экспоненциальной и дробной частей. Фактически, *comp* – это «большое» целое число со знаком. В то же время в выражениях *comp* полностью совместим с любыми другими вещественными типами. Наиболее подходящей

областью применения типа *comp* являются бухгалтерские расчеты.

Перечень основных процедур и функций, применимых к переменным вещественного типа, приведен в табл. 2.4.

Таблица 2.4. **Перечень функций с переменными вещественного типа**

Функция	Действие	Поясняющий пример
abs(x)	Модуль *x*	*abs(*-5,5*)*=5,5
arctan(x)	Арктангенс *x*	
cos(x)	Косинус угла *x*	
exp(x)	Экспонента в степени *x*	*exp(1)*=2,71…
frac(x)	Дробная часть числа *x*	*frac(*5,34*)*=0,34
int(x)	Целая часть числа *x*	*int(*5,34*)*=5
ln(x)	Натуральный логарифм *x*	*ln(1)*=0
pi	Число пи	*pi* =3.141565..
random	Псевдослучайное число, равномерно распределенное в диапазоне 0..1	
random(x)	Псевдослучайное число, распределенное в диапазоне 0..(*x-1)*	
sin(x)	Синус угла *x*	
sqr(x)	Квадрат *x*	*sqr(5)*=25
sqrt(x)	Корень квадратный *x*	*sqrt(*49*)*=7

Целые типы

Целые типы предназначены для задания целых чисел. В табл. 2.5 приводится названия целых типов, длина их внутреннего представления в байтах и диапазон возможных значений.

Таблица 2.5. **Целые типы данных и их параметры**

Название	Длина, байт	Диапазон значений
byte	1	0…255
shortint	1	-127...127
word	2	0…65535
integer	2	-32 768...32 767
longint	4	-2 147 483 648...2 147 483 647

Диапазон возможных значений целых типов зависит от их внутреннего представления, которое может занимать один, два или четыре байта.

Переменные целочисленного типа могут быть аргументами функций применяемых для вещественных переменных (табл. 2.4). В табл. 2.6 представлены дополнительные функции, применимые только к переменным целого типа.

Таблица 2.6. **Перечень функций с переменными целого типа**

Функция	Действие	Поясняющий пример (x=5)
dec(x)	Уменьшает значение x на 1	*dec(*x*)*=4
inc(x)	Увеличивает значение x на 1	*inc(*x*)*=6

Логический тип

Значениями логического типа может быть одна из предварительно объявленных констант *false* (ложь) или *true* (истина). Название типа при описании переменной ***boolean***.

Символьный тип

Значением символьного типа является множество всех символов компьютера. Каждому символу приписывается целое число в диапазоне 0...255. Это число служит кодом внутреннего представления символа. Для кодировки используется код ASCII (American Standard Code for Information Interchange - американский стандартный код для обмена информацией). Таблицу кодов можно найти в сети Интернет. Название типа при описании переменной символьного типа – ***char***.

К типу ***char*** применимы операции отношения, а также встроенные функции, приведенные в табл. 2.7.

Таблица 2.7. **Перечень функций для переменных символьного типа**

Функция	Действие	Поясняющий пример (x=5)
chr(x)	Возвращает символ по его коду	*chr(*71*)*=’G’
ord(c)	Возвращает код введенного символа	*ord(*‘G’*)*=71
upcase(x)	Возвращает прописную букву	*upcase(*‘a’*)*=’A’

2. К структурированным типам данных относятся: массивы, записи, множества и файлы. Они характеризуются множественностью образующих этот тип элементов, т.е. переменная или константа структурированного типа всегда имеет несколько компонентов. Каждый компонент, в свою очередь, может принадлежать структурированному типу. Записи, множества и файлы выходят за границы базового курса и в данном разделе не рассматриваются.

Массивы

Массивы в Паскале во многом схожи с матрицами в математике. Отличительная особенность массивов заключается в том, что все их компоненты должны быть одного типа, но не обязательно вещественного, как в матрицах. Переменная, являющаяся массивом, задается следующим образом:

<*переменная*>:***array*** [<*диапазон*>] ***of*** <*тип*>;

где

array, of	–	зарезервированные слова (*массив, из*);
<*диапазон*>	–	список из одного или нескольких индексных типов, разделенных запятыми;
<*тип*>	–	любой тип Турбо Паскаля.

Определить переменную как массив можно непосредственно при описании этой переменной, например:

```
var
  a: array [1..10] of real;
```

В приведенном примере задан массив, представляющий собой одномерную матрицу-строку из десяти дробных чисел. Двух или многомерные массивы записываются несколькими диапазонами, перечисляемыми через запятую. Например, объявление двухмерной прямоугольной матрицы размером 10×5 будет иметь вид

```
var
  b: array [1..10, 1..5] of real;
```


Обращение к элементу массива в программе производится по его номеру (номерам), например, a[1] или b[2,5].

3. Строковый тип широко используется для обработки текстов. Он во многом похож на одномерный массив символов ***array*** *[0..n]* ***of*** *char,* однако, в отличие от последнего, количество символов в строке-переменной может меняться от 0 до *n,* где *n* – максимальное количество символов в строке. Значение *n* задают при объявлении типа ***string****[n]*, которое может быть любой константой порядкового типа, но не больше 255. Паскаль разрешает не указывать *n,* в этом случае длина строки принимается максимально возможной, а именно *n=255*. Пример объявления строковой переменной:

```
var
  s: string [10];
  stroka: string;
```

Строка в Паскале трактуется как цепочка символов. К любому символу в строке можно обратиться точно так же, как к элементу одномерного массива, например, для строки *s*=’*Программа*’, *s*[2]=’*р*’.

К типу ***string*** применимы операции сложения «+», т.е. сцепление двух строк, а также встроенные функции, приведенные в табл. 2.8.

Таблица 2.8. **Перечень основных функций для переменных-строк**

Функция	Действие	Поясняющий пример*
length(s)	Возвращает длину строки	*length(*s1*)*=3
concat(s1,s2,..)	Возвращает строку, представляющую собой сцепление строк *s1,s2,..*	*concat(*s1,s2*)*= ’*программа*’
copy(s,ind,cnt)	Копирует из строки *s cnt* символов начиная с символа номер *ind*	*copy(*s2,1,4*)*= ‘грам’
pos(*subst,st*)	Отыскивает в строке *st* первое вхождение подстроки *subst* и возвращает номер позиции	*pos*(‘ра’,s2)=2 *pos*(‘ра’,s1)=0

* Примечание: *s1*=’*про*’; *s2*=’*грамма*’.

2.3.7. Совместимость типов

Паскаль – это типизированный язык. Он построен на основе строгого соблюдения концепции типов, в соответствии с которой все применяемые в языке операции определены только над операндами совместимых типов. При несовместимости типов возникают ошибки. Два типа считаются совместимыми, если оба они есть один и тот же тип.

В программе данные одного типа могут преобразовываться в данные другого типа. Такое преобразование может быть явным или неявным. При явном преобразовании типов используются вызовы специальных функций преобразования ***ord, trunc, round, chr***. Их описание приведено в табл. 2.9.

Таблица 2.9. **Функции преобразования типов**

Функция	Способ преобразования	Поясняющий пример
chr(x)	Преобразование целого числа x в символьный тип по таблице ASCII	*chr(*71*)*=’G’
ord(c)	Преобразование x в символа *c* в целое число по таблице ASCII	*ord(*‘G’*)*=71
round(x)	Преобразование дробного числа x до целого числа путем округления	*round(*1.5*)*=2 *round(*1.4*)*=1
trunc(x)	Преобразование дробного числа x до целого числа путем отбрасывания дробной части	*trunc(*3.99*)*=3

Неявное преобразование типов возможно в выражениях, составленных из вещественных и целочисленных переменных. Последние автоматически преобразуются к вещественному типу, и все выражение в целом приобретает вещественный тип. Например, при умножении целого числа 2 на вещественное число 1.5 результат будет иметь вещественный тип 3.0, несмотря на то, что дробной части в числе нет. Поэтому результат должен быть присвоен переменной вещественного типа, иначе возникнет ошибка.

2.3.8. Операции над переменными

В Турбо Паскале есть все четыре арифметические операции над переменными целого и вещественного типов:

«+» – сложение;
«–» – вычитание;
«*» – умножение;
«/» – деление вещественное;
« ***div*** » – деление целочисленное.

Наличие двух операций деления есть еще одно проявление основополагающего принципа Паскаля: программист должен явно подтверждать, какой тип результата он должен получить. Использование операции целочисленного деления ***div*** свидетельствует о том, что программист сознательно отбрасывает дробную часть результата. Например, 5/2=2.5, а 5 ***div*** 2=2.

Для данных целого типа в Паскале есть еще одна операция ***mod*** – получение остатка от целочисленного деления. Например:

5 ***mod*** 2 =1; 31 ***mod*** 16 = 15; 18 ***mod*** 3 =0.

Над символами и строками символов определена единственная операция – сцепление двух строк. Операция обозначается символом «+». Например,

s:=’Турбо’+’ ’+’Паскаль’; s=’Турбо Паскаль’.

Все остальные действия над строками и символами реализуются с помощью встроенных процедур и функций.

Над данными целого, вещественного и символьного типа, а так же над строками определены следующие операции отношения (сравнения):

«=» – равно;
«<>» – не равно;
«<» – меньше;
«>» – больше;
«<=» – меньше или равно;
«>=» – больше или равно.

В операциях сравнения должны участвовать однотипные операнды. Исключение сделано опять-таки в отношении целых и вещественных типов, которые могут сравниваться друг с другом. Результат применения операции отношения к любым операндам имеет тип *boolean*.

В Турбо Паскале определены следующие логические операции:

not – логическое НЕ;
or – логическое ИЛИ;
and – логическое И;
xor – логическое исключающее ИЛИ.

Логические операции применимы к операндам целого и логического типов. Если операнды – целые числа, то результат логической операции есть тоже целое число. Логические операции над логическими данными дают результат логического типа.

При вычислении выражений любого типа приоритет вычислений определяется расставленными скобками, а при их отсутствии – в соответствии с таблицей приоритета (табл. 2.10).

Таблица 2.10. **Приоритет операций над переменными**

Приоритет	Операция
1	***not***
2	*, /, ***div***, ***mod***, ***and***
3	+, - , ***or***, ***xor***
4	=, <>, >, >=, <=, <, <=

2.3.9. Оператор присваивания

Для вычисления отношения введенных чисел используется один из основных операторов Турбо Паскаля – оператор присваивания. В его левой части указывается имя переменной, правая часть представляет собой выражение того же типа, что и переменная. Пара символов «:=», связывающая левую и правую

части оператора присваивания, означает «присвоить значение». В операторах присваивания Турбо Паскаля всегда используются символы «:=», в то время как при описании констант - одиночный символ «=». С точки зрения синтаксиса языка, два символа «:=» рассматриваются как один специальный символ и обязательно пишутся слитно. Пример:

```
y:=3*x+2;
```

В этом примере переменная x умножается на 3 и к значению прибавляется 2. Итоговое значение присваивается переменой y.

2.3.10. Условный оператор

Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Таким образом, условный оператор – это средство ветвления вычислительного процесса.

Структура условного оператора имеет следующий вид:

If <*условие*> ***then*** <*оператор1*> ***else*** <*оператор2*>;

где

If, then, else – зарезервированные слова *(если, то, иначе)*;
<*условие*> – произвольное выражение логического типа;
<*оператор1*>, <*оператор2*> – любые операторы языка Турбо Паскаль.

Условный оператор работает по следующему алгоритму. Вначале вычисляется условное выражение <*условие*>. Если результат есть *true* (истина), то выполняется <*оператор1*>, а <*оператор2*> пропускается; если результат есть *false* (ложь), наоборот, <*оператор1*> пропускается, а выполняется <*оператор2*>. Пример:

$$f(x)=\begin{cases}3x+1, \text{если } x\le 0\\ x^2+1, \text{если } x>0\end{cases}$$

```
if x<=0 then f_x:=3*x+1
else f_x:=sqr(x)+1;
```

Часть ***else*** <*оператор2*> условного оператора может отсутствовать. Тогда при значении *true* условного выражения выполняется <*оператор1*>, в противном случае этот оператор пропускается. Пример:

$f(x) = \ln x$, *если* $x > 0$. **if** x>0 **then** f_x:=*ln*(x);

Поскольку любой из операторов <*оператор1*> и <*оператор2*> может быть любого типа, в том числе и условным, а в то же время не каждый из «вложенных» условных операторов может иметь часть ***else*** <*оператор2*>, то возникает неоднозначность трактовки условий. Эта неоднозначность в Паскале решается образом: любая встретившаяся часть ***else*** соответствует ближайшей к ней части ***then*** условного оператора. Пример:

$$f(x) = \begin{cases} 3x + 1, \text{если } x \le -2, \\ x, \text{если } x > -2 \text{ и } x \le 0, \\ \ln x, \text{если } x > 0 \end{cases}$$

if x<=-2 **then** f_x:=3*x+1
else if x<=0 **then** f_x:=x
else f_x:=*ln*(x);

2.3.11. Составной оператор

Составной оператор – это последовательность произвольных операторов программы, заключенная в операторные скобки – зарезервированные слова «***begin ... end;***». Составные операторы - важный инструмент Паскаля, дающий возможность писать программы по современной технологии структурного программирования (без операторов перехода ***goto***).

Язык Паскаль не накладывает никаких ограничений на характер операторов, входящих в составной оператор. Среди них могут быть и другие составные операторы – Паскаль допускает произвольную глубину их вложенности.

Рассмотрим использование составного оператора на примере условного оператора. При составлении части программы, содержащей условный оператор, после зарезервированных слов ***then*** и ***else*** стоит по одному оператору. Однако в большинстве

случаев внутри каждого блока приходится выполнять не одно действие, а сразу несколько. Пример:

если $x > 0$, $y1 = \ln x$, $y2 = x - 1$

if x>0 **then** y1:=*ln*(x);
y2:=x-1;

Если написать программу, как показано выше, то вычисление переменной y1 будет производится с учетом условия x>0, а переменной y2 – без. То есть второй оператор присваивания не будет находится внутри условного оператора. Для того чтобы условие применялось для обеих переменных, необходимо применить составной оператор:

если $x > 0$, $y1 = \ln x$, $y2 = x - 1$

if x>0 **then begin**
y1:=*ln*(x); y2:=x-1;
end;

2.3.12. Операторы цикла

В языке Турбо Паскаль имеются три различных оператора, с помощью которых можно запрограммировать повторяющиеся фрагменты программ.

Счетный оператор цикла for имеет такую структуру:

for <*пар*>:=<*нач_знач*> ***to*** <*кон_знач*> ***do*** <*оператор*>;

где

for, to, do	–	зарезервированные слова *(для, до, выполнить);*
<*пар*>	–	параметр цикла – переменная любого целого типа;
<*нач_знач*>	–	начальное значение;
<*кон_знач*>	–	конечное значение;
<*оператор*>	–	любой оператор языка Паскаль.

При выполнении оператора ***for*** вначале вычисляется выражение <*нач_знач*> и осуществляется присваивание <*пар_цик*>:=<*нач_знач*>. После этого циклически повторяется:

1. Проверка условия <*пар*> **<=** <*кон_знач*>; если условие не выполнено, оператор ***for*** завершает свою работу.

2. Выполнение оператора <*оператор*>;
3. Наращивание переменной <*пар*> на единицу.

Отметим два обстоятельства. Во-первых, условие, управляющее работой оператора ***for***, проверяется перед выполнением оператора <*оператор*>. Если условие не выполняется в самом начале работы оператора ***for***, исполняемый оператор не будет выполнен ни разу. Другое обстоятельство – шаг наращивания параметра цикла строго постоянен и равен (+1). Существует другая форма оператора:

for <*пар*>:=<*нач_знач*> ***downto*** <*кон_знач*> ***do*** <*оператор*>;

Замена зарезервированного слова ***to*** на ***downto*** означает, что шаг наращивания параметра цикла равен (-1).

Ниже представлен пример использовании цикла ***for*** для вычисления суммы *s* и произведения *p* всех чисел от 10 до 20 включительно:

```
x:=0; p:=1;
for k:=10 to 20 do
      begin
      x:=x+k; y:=y*k;
      end;
```

Два других оператора циклов лишь проверяют условие выполнения или повторения цикла, но не связаны с изменением счетчика цикла.

Оператор цикла «пока» с предусловием имеет форму

while <условие> ***do*** <оператор>;

где

while, do	–	зарезервированные слова *(пока выполняется условие, делать)*;
<*условие*>	–	выражение логического типа;
<*оператор*>	–	любой оператор языка Паскаль.

Если выражение <*условие*> имеет значение *true*, то выполняется <*оператор*>**,** после чего вычисление выражения <*условие*> и его проверка повторяются. Если <*условие*> имеет значение *false*, оператор ***while*** прекращает свою работу.

В качестве примера решим ту же задачу, что и в примере для цикла ***for***:

```
x:=0; p:=1; k:=10;
while k<=20 do
      begin
      x:=x+k; y:=y*k;
      k:=k+1;
      end;
```

Оператор цикла «пока» с постусловием имеет форму:

repeat <*тело_цикла*> ***until*** <*условие*>;

где

repeat, until	–	зарезервированные слова *(повторять до тех пор, пока не будет выполнено условие)*;
<*тело_цикла*>	–	произвольная последовательность операторов;
< *условие*>	–	выражение логического типа.

Операторы <*тело_цикла*> выполняются хотя бы один раз, после чего вычисляется выражение <*условие*>. Если его значение есть *false*, операторы <*тело_цикла*> повторяются, в противном случае оператор «***repeat ... until***» завершает свою работу. В качестве примера решим ту же задачу, что и в примере для цикла ***for***:

```
x:=0; p:=1; k:=10;
repeat
      x:=x+k; y:=y*k;
      k:=k+1;
until k>20;
```

Для гибкого управления циклическими операторами ***for***, ***while*** и ***repeat*** в состав Паскаля включены две процедуры: *break* – реализует немедленный выход из цикла; действие процедуры заключается в передаче управления оператору, стоящему сразу за концом циклического оператора; *continue* – обеспечивает досрочное завершение очередного прохода цикла; эквивалент передачи управления в самый конец циклического оператора.

2.3.13. Оператор перехода

Рассмотренных выше операторов вполне достаточно для написания программ любой сложности. В этом отношении наличие в языке операторов перехода кажется излишним. Более того, современная технология структурного программирования основана на принципе «программировать без goto». Считается, что злоупотребление операторами перехода затрудняет понимание программы, делает ее запутанной и сложной в отладке. Тем не менее в некоторых случаях использование операторов перехода может упростить программу.

Оператор перехода имеет вид

goto <*метка*>;

где

goto – зарезервированное слово *(перейти на метку)*;
<*метка*> – имя метки.

Метка в Турбо Паскале – это произвольный идентификатор, позволяющий именовать некоторый оператор программы и таким образом ссылаться на него. Метка располагается непосредственно перед помечаемым оператором и отделяется от него двоеточием. Оператор можно помечать несколькими метками, которые в этом случае отделяются друг от друга двоеточием. Перед тем как появиться в программе, метка должна быть описана в блоке ***label***. В качестве примера приведем программу, реализующую блок повторения без операторов цикла (задача в примере для цикла ***for***):


```
label loop1,loop2;
begin
    x:=0; p:=1; k:=10;
    loop1: if k> 20 then goto loop2;
    x:=x+k; y:=y*k;
    k:=k+1;
    goto loop1;
    loop2:
end.
```

2.3.14. Разделитель операторов

Как вы уже, наверное, успели отметить, для разделения отдельных операторов используется точка с запятой «;». Точка с запятой – аналог точки в обычных языках, означающих окончание одного высказывания и начало другого. Использование обычного разделителя в виде пробела невозможно, так как некоторые операторы, например, ***mod*** или ***div***, используют его для отделения от идентификаторов.

Точка с запятой требуется после каждого оператора в блоке, за исключением последнего. Для исключения ошибок разделитель «;» можно ставить после каждого оператора, в том числе последнего, за одним исключением. Прямо перед ***else*** вы абсолютно не можете поставить точку с запятой, иначе это приведет к появлению ошибки при компиляции.

2.3.15. Вызов процедур и функций

Процедурой в Турбо Паскале называется особым образом оформленная часть программы, имеющая собственное имя. Упоминание этого имени в тексте программы приводит к активизации процедуры и называется ее вызовом. Сразу после активизации процедуры начинают выполняться входящие в нее операторы. После выполнения последнего из них, управление возвращается обратно в основную программу, и выполняются опе-

раторы, стоящие непосредственно за оператором вызова процедуры.

Для обмена информацией между основной программой и процедурой используется один или несколько параметров вызова. Если они есть, то они перечисляются в круглых скобках за именем процедуры и вместе с ним образуют оператор вызова процедуры.

<имя_процедуры>(*<параметр1>*,*<параметр2>*,…);

Пример использования процедуры вывода строки *Writeln*:

Writeln(‘Значение функции равно ’,s);

Функция отличается от процедуры тем, что результат ее работы возвращается в виде значения этой функции. Вызов функции, таким образом, можно осуществлять в составе выражений везде, где возможно использование выражений (в операторе присваивания, в операторе вывода и т.д.), например:

<переменная>:=*<имя_функции>*(*<фактические параметры>*);

Ниже приведены примеры вызова функции *round*(), округляющей значение, стоящее в скобках, до целого числа, в операторе присваивания и условном операторе:

x:=*round*(x/5);
if *round*(y/5)<2 ***then*** z:=z+1;

2.3.16. Процедуры ввода и вывода информации

Процедуры ввода и вывода предназначены для организации взаимодействия программы с пользователем. Перед выполнением определенных действий или вычислений необходимо запросить у пользователя исходные данные. После расчетов для пользователя необходимо вывести результат. Эти задачи решают с использованием встроенных процедур.

Работа программ, написанных на Турбо Паскале, предусмотрена в операционной системе MS DOS, в которой ввод–вывод информации осуществляется в текстовом режиме с клавиатуры. Привычный способ взаимодействия, примененный в операционной системе Windows на основе использования мыши, в этом случае не предусмотрен. Для знакомства с операционной системой MS DOS с консольной системой ввода–вывода информации в операционной системе Windows существует интерпретатор командной строки. Для этого в меню «Пуск» необходимо выбрать подпункт «Выполнить», набрать в нем «command» и нажать «Enter». Перед вами интерпретатор MS DOS.

Для вывода информации в текстовом режиме предусмотрены две процедуры:

Write(*<список_вывода>*);
Writeln(*<список_вывода>*);

В данных процедурах под параметром *<список_вывода>* подразумевается список переменных и констант, перечисляемых через запятую. Пример:

Writeln(‘Значение функции f(x)=’,f_x,’ при x=’,x);

При выводе информации текстовые выражения, заключенные в апострофы, на экране будут представлены без изменений, а вместо переменных *f_x* и *x* программой будут подставлены их значения. Следует обратить внимание, что при выводе значений переменных в операторе *Writeln* даются предварительные текстовые пояснения вида «Значение функции f(x)=». Сделано это из-за того, что конечным пользователем является не программист, создавший программу и знающий, что выводит программа, а некий пользователь. И если на экран вывести только значения переменных, ему будет не ясно, что за значения представлены программой.

Различие в процедурах *Write* и *Writeln* заключается в том, что после вывода данных оператором *Write* курсор остается за последним выведенным символом. Оператор *Writeln* после вывода данных переводит курсор на первую позицию следующей строки. Процедура *Write* обычно используется, когда надо вывести для пользователя сообщение на экран, после чего получить данные, не переводя курсор на новую строку. Например, выводим на экран «Введи число:» и ждем ввода в этой же стро-

ке. Процедура *Writeln* обычно используется для переноса следующего выводимого текста на новую строку, что улучшает восприятие выводимой информации.

При выводе данных можно задавать формат вывода. Для этого после имени переменной ставится знак двоеточие «:», например: *Writeln*(a:5:2); - при выводе значения переменной вещественного типа отводится 5 позиций (включая отрицательный знак и точку), из них 2 позиции отводится для вывода цифр в дробной части. При выводе значения переменной целого типа задается количество позиций для числа (включая отрицательный знак), например: *Writeln*(i:8).

Для ввода данных с клавиатуры используются процедуры:

Read(<*список_ввода*>);
Readln(<*список_ввода*>);

В данных процедурах под параметром <*список_ввода*> подразумевается список переменных, для которых вводятся значения, перечисляемых через запятую. Перед вводом данных с клавиатуры желательно вывести на экран поясняющее сообщение. Для этого в программу следует включить оператор вывода, например: *Write*('введите a='). Затем вставить оператор ввода *Readln*(a). При выполнении программы на экране появится надпись: «введите a=». Оператор Readln(a); будет ждать ввода данных до нажатия пользователем клавиши «Enter». По причине необходимости давать пояснения по вводимым данным рекомендуется использовать процедуры ввода с одной переменной, а не со списком из нескольких переменных.

Различие в процедурах *Read* и *Readln* аналогичное процедурам *Write* и *Writeln*. Преимущественно используется вторая форма записи оператора ввода.

2.3.17. Пример программы

В качестве примера создадим программу для решения задачи, приведенной в разделе 2.3. Программу удобнее составлять по разработанному алгоритму, заменяя блоки на соответствующие операторы Паскаль. Такой подход представлен на рис. 2.18.

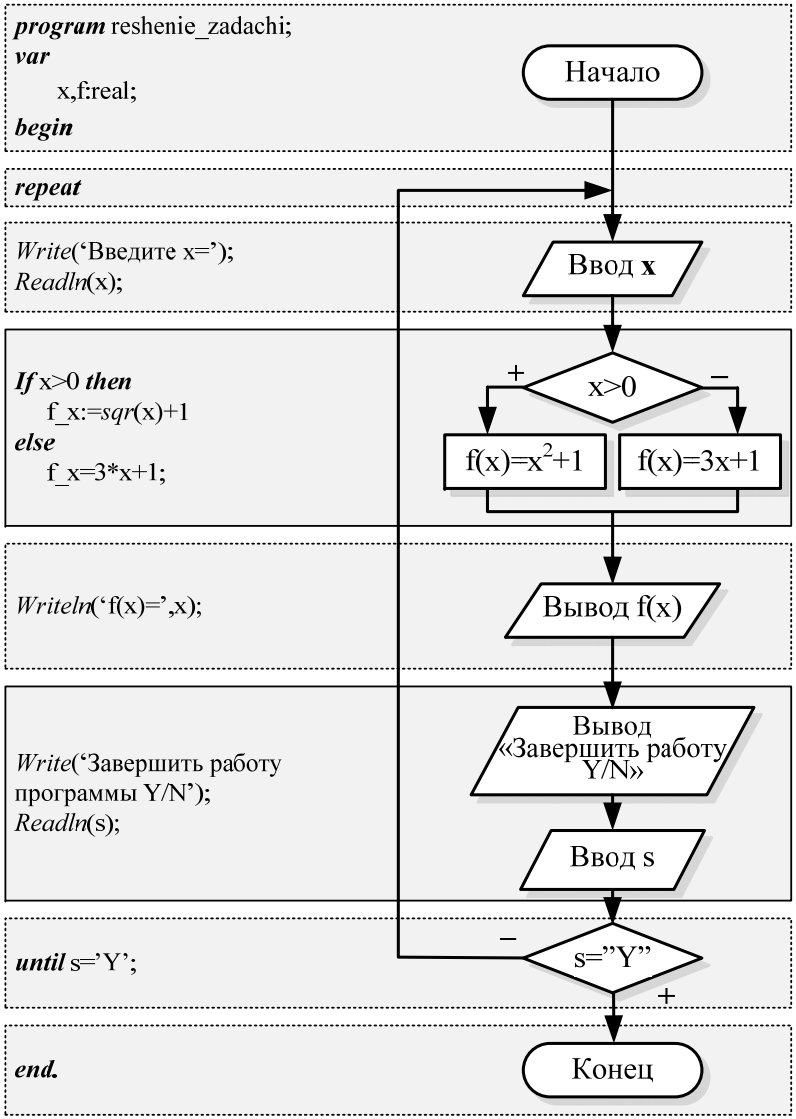


Рис. 2.18. Программа на языке программирования Паскаль, выполненная на базе алгоритма

РАЗДЕЛ ТРЕТИЙ
Варианты заданий контрольной работы

До начала аудиторных занятий студентом должна быть выполнена контрольная работа. Решения заданий контрольной работы должны быть представлены в письменной форме в тетради в клетку. Работы, выполненные в печатной форме, к рассмотрению не принимаются.

Выбор варианта контрольного задания производится по последним ***двум*** цифрам зачетной книжки. Схема определения варианта задания представлена на рис. 3.1.

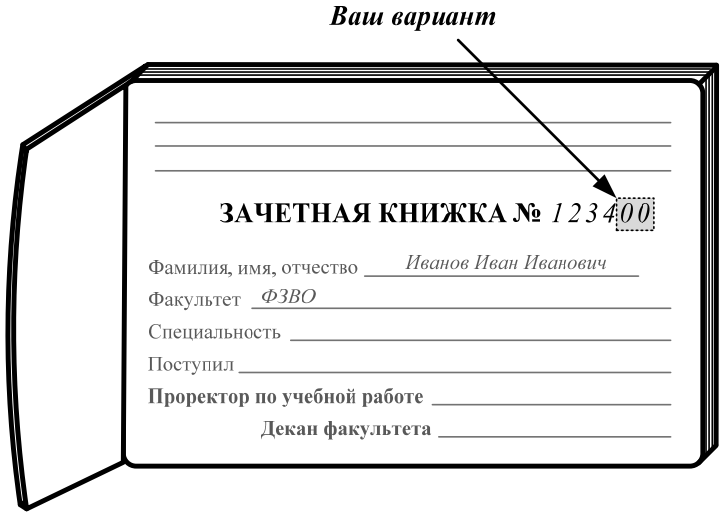


Рис. 3.1. Схема определения варианта контрольного задания

Контрольная работа включает в себя теоретическую и практическую части. Теоретическая часть включает в себя два вопроса. Для ее выполнения следует изучить соответствующие разделы рекомендуемой литературы и в краткой лаконичной форме изложить материал в контрольной работе. Средний объем ответа на один теоретический вопрос составляет 1–2 страницы

рукописного текста. Варианты теоретических заданий представлены в табл. 3.1.

Практическая часть включает в себя два раздела: системы счисления и программирование. Варианты заданий первого раздела практической части представлены в табл. 3.2 и заданий второго раздела практической части – в табл. 3.3 настоящего учебно-методического пособия. Теория и методика выполнения практической части работы представлены во втором разделе учебно-методического пособия.

Первый раздел практической части посвящен переводу чисел между различными системами счисления. Он состоит из девяти заданий.

1. Перевести число из десятичной системы счисления в двоичную систему счисления.
2. Перевести число из двоичной системы счисления в десятичную систему счисления.
3. Перевести число из десятичной системы счисления в шестнадцатеричную систему счисления.
4. Перевести число из шестнадцатеричной системы счисления в десятичную систему счисления.
5. Перевести число из десятичной системы счисления в восьмеричную систему счисления.
6. Перевести число из восьмеричной системы счисления в десятичную систему счисления.
7. Перевести число из двоичной системы счисления в восьмеричную и шестнадцатеричную системы счисления.
8. Перевести число из шестнадцатеричной системы счисления в двоичную и восьмеричную системы счисления.
9. Перевести число из восьмеричной системы счисления в двоичную и шестнадцатеричную системы счисления.

Во втором разделе практической части следует составить программу для решения определенной прикладной задачи. В работе необходимо составить блок-схему алгоритма решения задачи, и написать программу на основе блок-схемы с использованием языка программирования Turbo Pascal.

Таблица 3.1. **Контрольные задания. Теоретическая часть**

Вариант	Вопрос №1	Вопрос №2
00	Что такое информация? Что такое кодирование?	Оператор цикла с предусловием While Do в языке программирования Паскаль.
01	Что означает динамический характер информации? Основные операции с данными.	Оператор цикла с постусловием Repeat-Until в языке программирования Паскаль.
02	Как объяснить требование адекватности методов.	Оператор цикла For Do в языке программирования Паскаль.
03	Диалектический характер взаимодействия данных и методов. Формализация данных.	Условный оператор в языке программирования Паскаль
04	Свойства информации. Дайте определение понятию «Фильтрация данных».	Понятие «топология локальных сетей». Перечислите и охарактеризуйте типы конфигурации сети.
05	Объективность и субъективность информации. Сортировка данных.	Компьютерные вирусы и их свойства.
06	Полнота информации. Дайте определение понятию «Архивация данных».	Перечислите и дайте характеристики программно-техническим методам обнаружения вирусов.
07	Достоверность информации. Дайте определение понятию «Защита данных».	Методы описания цветов в компьютерной графике – цветовые модели. Способы получения цветовых оттенков на экране монитора и принтере.
08	Адекватность информации. Дайте определение понятию «Защита данных».	Преимущества и способы работы со слоями в векторном редакторе.
09	Доступность информации. Дайте определение понятию «Преобразование данных».	Приемы редактирования объектов в векторном редакторе (вставка, копирование, удаление и т.д.).
10	Информационные процессы и системы.	Понятие алгоритма и его свойства.

Продолжение табл. 3.1

Вариант	Вопрос №1	Вопрос №2
11	Типы компьютерной памяти. Их предназначение. Характеристики.	Языки программирования высокого уровня и низкого уровня. Различия. Достоинства и недостатки.
12	Преобразование аналоговой информации в цифровую. Назначение. Способы.	Объектно-ориентированное программирование. Особенности. Достоинства и недостатки.
13	Классификация запоминающих устройств. Основные характеристики.	Структурное программирование. Особенности. Достоинства и недостатки.
14	Дайте характеристику основным системам обработки текстов.	Основные конструкции языка Pascal.
15	Способы передачи информации.	Принципы разработки и анализа алгоритмов.
16	Системы компьютерной графики.	Алфавит языка программирования Turbo Pascal.
17	Информационные ресурсы и технологии.	Основные понятия баз данных.
18	Редакторы формул в текстовых процессорах MS Word, Open Office. Назначение. Порядок набора формул.	Настройки рабочего листа в графическом редакторе Visio.
19	Порядок построения графиков функции одной и двух переменных в MS Excel.	Что относится к примитивам в графическом редакторе? Способы их редактирования.
20	Что такое относительные и абсолютные адреса ячеек в MS Excel? Примеры их использования.	Дайте определение типов полей, используемых в базе данных MS Access.
21	Что такое графический редактор? Классификация графических редакторов.	Что собой представляет база данных? Структура базы данных.
22	Из каких структурных элементов состоит файл электронной таблицы?	Программирование арифметических выражений в языке Turbo Pascal. Приведите пример.

Продолжение табл. 3.1

Вариант	Вопрос №1	Вопрос №2
23	Порядок записи формул в MS Excel. Примеры.	Программирование разветвляющихся структур в языке Turbo Pascal. Приведите пример.
24	Прикладное программное обеспечение: проблемно-ориентированные, пакеты общего назначения и интегрированные пакеты.	Оператор цикла с параметром в языках программирования высокого уровня. Приведите пример.
25	Инструментальное программное обеспечение. Дать определение и перечислить системы программирования программ с использованием визуальных средств.	Оператор цикла с предусловием в языках программирования высокого уровня. Приведите пример.
26	Дать определение и описание структуры программного обеспечения ПК.	Оператор цикла с постусловием в языках программирования высокого уровня. Приведите пример.
27	Классификация операционных систем. Достоинства и недостатки.	Структура данных в языке Turbo Pascal. Приведите пример.
28	Что такое компьютер? Основные устройства. Классификация компьютеров по типоразмерам.	Процедуры и функции в языке Turbo Pascal. Приведите пример.
29	Проприетарное, свободное и открытое программное обеспечение.	Системы управления базами данных. Определение. Назначение.
30	Электронная подпись и электронный сертификат.	Трехуровневая архитектура описания баз данных.
31	Линейные и табличные структуры данных.	Концепции проектирования баз данных.
32	Единицы представления данных. Единицы хранения данных.	Сетевая модель базы данных. Приведите пример.

Продолжение табл. 3.1

Вариант	Вопрос №1	Вопрос №2
33	Понятие о файловой структуре. Какие файловые системы существуют?	Иерархическая модель базы данных. Приведите пример.
34	Состав программного обеспечения.	Реляционная модель базы данных. Пример.
35	Классификация программного обеспечения. Прикладное программное обеспечение.	Понятие класса, свойства, методов в объектно-ориентированное программировании.
36	Средства антивирусной защиты. Назначение. Принципы функционирования. Примеры.	Концепции инкапсуляции, полиморфизма, наследия. Определения и основные принципы.
37	Информационное обеспечение.	Назовите методы проектирования алгоритмов и дайте их краткое описание.
38	Шестнадцатеричная система счисления. Представление отрицательных и дробных чисел.	Нисходящий и восходящий методы проектирования алгоритмов. Приведите примеры.
39	Что собой представляет база данных? Преимущества и недостатки обработки данных на основе технологии баз данных.	Модульный метод проектирования алгоритмов и программ.
40	Перечислите и дайте краткую характеристику типам полей таблиц MS Access.	Операции над структурами данных. Приведите примеры.
41	Защита от несанкционированного доступа. Основные методы. Эффективность.	Понятие структуры данных программ. Приведите примеры.
42	Перечислите компоненты системы управления базами данных (СУБД) и дайте их описание.	Описание и вызов процедур и функций в языке программирования Turbo Pascal.
43	В чем заключается концепции проектирования баз данных?	Процедуры и функции в языках программирования. Различия. Примеры использования.

Продолжение табл. 3.1

Вариант	Вопрос №1	Вопрос №2
44	Сетевое оборудование локальной сети. Назначение. Краткая характеристика.	Основные положения концепции модульного программирования.
45	Перечислите и дайте краткое описание видов локальных сетей.	Определения модуля в программировании. Разновидности модулей.
46	Протоколы TCP и UDP. Назначение. Достоинства и недостатки.	Встроенные арифметические и логические функции языка программирования Turbo Pascal.
47	Назовите основные протоколы передачи данных и дайте их краткую характеристику.	Общая структура программы на языке программирования высокого уровня.
48	Локальные (LAN) и глобальные (WAN) сети.	Перечислите и опишите служебные слова языка Turbo Pascal.
49	Коммуникационное оборудование глобальных сетей. Назначение. Краткая характеристика.	Арифметические выражения в Turbo Pascal. Перечислите операции и соответствующие знаки, используемые в арифметических выражениях.
50	Принципы защиты информации. Способы защиты от несанкционированного доступа.	Перечислите и опишите стандартные функции языка Pascal для работы с символами и строками.
51	Классификация программного обеспечения. Служебное программное обеспечение.	Перечислите основные правила программирования арифметических выражений в языке Turbo Pascal**.** Приведите пример.
52	Компьютерные вирусы. Классификация. Основные угрозы.	Опишите процедуру ввода и вывода информации в языке Pascal**.** Приведите пример ввода и вывода.
53	Административно-технологические методы защиты корпоративной интрасети.	Программирование вложенных ветвлений. Приведите пример.

Продолжение табл. 3.1

Вариант	Вопрос №1	Вопрос №2
54	Коммерческое, свободно распространяемое и частично-распространяемое программное обеспечение. Отличия. Примеры.	Простой и составной операторы в языке Turbo Pascal. Порядок выполнения операций составного оператора.
55	Растровая и векторная графики. Отличия. Достоинства и недостатки. Примеры программ.	Линейные процессы вычислений. Достоинства и недостатки. Пример линейных программ.
56	Совместимость и взаимодействие антивирусных программ.	Алгоритм разветвляющейся структуры. Полная, неполная форма организации ветвления.
57	Каково назначение процессора (CPU) и каковы выполняемые им операции?	Оператор вариантов Case в языке программирования Pascal. Синтаксис. Пример использования.
58	Назначение компьютеров. Их классификация по назначению и уровню специализации.	Создание маркированных, нумерованных и других типов списков MS Word. Форматирование списков.
59	Относительные и абсолютные адреса ячеек в электронной таблице. Назначение. Отличия. Пример использования.	Построение изображений с использованием библиотек в векторном редакторе MS Visio или AutoCad.
60	Инструментальные программные средства специального и общего назначения. Примеры.	Преобразование аналоговой информации в цифровую информацию.
61	Системное программное обеспечение. Примеры их применения.	Общая схема системы передачи информации.
62	Чем отличаются понятия "информация" и "данные"? Основные свойства информации.	Назначение интерпретатора и компилятора. Примеры служебных интерпретаторов и компиляторов.
63	Состав вычислительной системы.	Свойства реляционной модели базы данных.

Продолжение табл. 3.1

Вариант	Вопрос №1	Вопрос №2
64	Этапы создания баз данных. Разработка технического задания и структуры БД.	Современные средства создания программ и среды программирования.
65	Опишите типы ключевых полей баз данных: счетчик, простой и составной ключи. Примеры.	Алгоритм и его свойства. Блок-схемы, используемые для изображения алгоритмов.
66	Какие связи между таблицами можно организовать в базах данных? Примеры связей.	Системное программное обеспечение. Программы диагностики работоспособности компьютера.
67	Типы запросов в базах данных. Приведите пример их использования.	Системное программное обеспечение. Программы обслуживания дисков.
68	Перечислите и дайте характеристику видам компьютерной графики.	Системное программное обеспечение. Программы архивирования данных.
69	Правовое обеспечение информационной безопасности.	Современные аппаратные средства растровой графики. Устройства ввода.
70	Основные принципы развития компьютерных систем.	Создание таблицы в MS Word. Способы редактирования ячейки, группы ячеек, строки, столбца.
71	Какие основные параметры характеризуют производительность компьютера?	Порядок создания математических формул в объекте Microsoft Equation. Приведите пример.
72	Назначение операционной системы. Наиболее распространенные операционные системы.	Общие правила написания идентификаторов в Pascal. Примеры.
73	Система управления базами данных (СУБД). Организация структуры хранения данных.	Перечислите типы полей базы данных. Для чего применяется поле МЕМО?
74	Протоколы передачи данных сети Интернет. Назначение, характеристика.	Арифметико-логическое устройство персонального компьютера. Назначение, принцип работы.

Продолжение табл. 3.1

Вариант	Вопрос №1	Вопрос №2
75	Кодирование графической информации. Цветовые модели RGB и CMYK.	Классификация компьютерных сетей по территориальному признаку и топологии.
76	Кодирование звуковой информации. Цифро-аналоговое и аналогово-цифровое преобразование.	Опишите процессы исследования, анализа, программирования и тестирования при создании программ.
77	Классификация персональных компьютеров.	Основные направления применения сети Интернет.
78	Единицы представления данных.	Адресация в сети Интернет. Доменная система имен.
79	Архитектура СУБД.	Понятие интрасеть (intranet) и ее характеристики.
80	Технические устройства и системы сбора и обработки информации.	Что такое сервер? Виды серверов.
81	Количество информации как мера уменьшения неопределенности знаний.	Чем отличается работа цикла с постусловием от работы цикла с предусловием?
82	Опишите алгоритм создания оглавления в редакторе MS Word(OpenOffice).	Средства разработки программ в современных системах программирования.
83	Основные сервисы Интернета.	Типы переменных в Turbo Pascal.
84	Перечислите и дайте характеристику типов поисковых систем сети Internet.	Средства работы с векторной графикой. Примеры. Достоинства и недостатки.
85	Правовая охрана программ и GNU GPL.	Автоматизация ввода информации в компьютер.
86	Программное обеспечение с открытым кодом (Open source). Приведите пример.	Средства обработки табличной информации. Какие задачи они позволяют решить?
87	Основные математические и логические функции в Excel.	Структура программ на языке Turbo Pascal. Приведите пример.
88	Архитектура ЭВМ. Принципы фон Неймана.	Назначение стилей в редакторе MS Word.

Окончание табл. 3.1

Вариант	Вопрос №1	Вопрос №2
89	Арифметические и логические выражения в языке Turbo Pascal.	Шаблоны документов в редакторе MS Word.
90	Передача информации по сети. Архитектура "Клиент - Сервер".	Мультимедийная презентация. Структура слайда. Форматирование слайдов.
91	Информационно-поисковые системы Internet. Их классификация.	Классификация баз данных по используемой модели. Сетевые базы данных.
92	Типы запросов в MS Access. Отличия в создании и выполнении.	Сервер, виды серверов, в зависимости от выполняемых функций.
93	Этапы решения задачи на компьютере.	Процедура поиска информации в сети Интернет.
94	Отчеты в базе данных. Их назначение в MS Access .	Основное и дополнительное оборудование локальной вычислительной сети. Назначение.
95	Операции с массивами в табличном процессоре MS Excel.	Центральный процессор, его назначение и характеристики.
96	Построение различных типов диаграмм по табличным данным в MS Excel.	Назначение, основные функции и основные характеристики видеокарт и мониторов.
97	Магистрально-модульный принцип построения компьютера.	Разработка алгоритмов методом последовательной детализации**.**
98	Двоичная система счисления. Представление отрицательных и дробных целых чисел.	Этапы загрузки операционной системы.
99	Системы счисления. Назначение перевода чисел между различными системами счисления.	Операторы присваивания и сравнения в Pascal. Примеры использования.

Таблица 3.2. **Контрольные задания. Практическая часть. Первый раздел**

Вариант	Номер задания								
	№1	№2	№3	№4	№5	№6	№7	№8	№9
00	2948_{10}	100011110_{2}	14371_{10}	$E8E_{16}$	7329_{10}	477_{8}	111010100101_{2}	$E1F8_{16}$	2701_{8}
01	3326_{10}	110010111_{2}	14050_{10}	$E5B_{16}$	11980_{10}	557_{8}	110000111110_{2}	$9FC9_{16}$	1752_{8}
02	3865_{10}	111110010_{2}	12912_{10}	$7E4_{16}$	7781_{10}	552_{8}	100011100010_{2}	$A21C_{16}$	3326_{8}
03	5019_{10}	100011110_{2}	11823_{10}	$8F2_{16}$	7514_{10}	510_{8}	110110101010_{2}	$AB80_{16}$	6002_{8}
04	4634_{10}	100100111_{2}	11655_{10}	$BB7_{16}$	8488_{10}	353_{8}	111100111100_{2}	$288D_{16}$	7541_{8}
05	4762_{10}	100101001_{2}	15180_{10}	$9D0_{16}$	11939_{10}	450_{8}	100000010010_{2}	$579B_{16}$	6777_{8}
06	4719_{10}	101000001_{2}	13474_{10}	$9EB_{16}$	7013_{10}	474_{8}	110010011000_{2}	5232_{16}	1320_{8}
07	4025_{10}	101100100_{2}	13105_{10}	$31C_{16}$	9437_{10}	550_{8}	100100011111_{2}	$6BDA_{16}$	4025_{8}
08	5590_{10}	100000101_{2}	11759_{10}	$9F8_{16}$	7847_{10}	232_{8}	101010110001_{2}	$F893_{16}$	2124_{8}
09	2402_{10}	111010101_{2}	12207_{10}	$D9A_{16}$	7153_{10}	215_{8}	101100010010_{2}	$87A1_{16}$	2335_{8}
10	2840_{10}	101001011_{2}	11527_{10}	288_{16}	11499_{10}	403_{8}	100110110101_{2}	$8A04_{16}$	2511_{8}
11	5360_{10}	101101010_{2}	14691_{10}	$F8A_{16}$	9284_{10}	477_{8}	110111101110_{2}	9368_{16}	5466_{8}
12	3812_{10}	111011010_{2}	12168_{10}	423_{16}	8803_{10}	427_{8}	111100000010_{2}	4061_{16}	1375_{8}
13	2370_{10}	100001001_{2}	15540_{10}	$B42_{16}$	8963_{10}	561_{8}	110110010110_{2}	$AF7F_{16}$	1272_{8}
14	5163_{10}	100011000_{2}	15885_{10}	$6BD_{16}$	8910_{10}	622_{8}	110000010110_{2}	$6D82_{16}$	1636_{8}
15	2898_{10}	101010000_{2}	13770_{10}	993_{16}	9370_{10}	522_{8}	101110011111_{2}	$76E1_{16}$	4522_{8}
16	3299_{10}	100101100_{2}	14475_{10}	$A1F_{16}$	11326_{10}	607_{8}	101110111010_{2}	$2FAE_{16}$	2431_{8}
17	2540_{10}	111101011_{2}	14771_{10}	566_{16}	7341_{10}	732_{8}	111001011000_{2}	$C2BC_{16}$	2643_{8}
18	5814_{10}	100011010_{2}	10141_{10}	$7AF_{16}$	11560_{10}	446_{8}	110101000011_{2}	$950B_{16}$	3215_{8}
19	5702_{10}	110110001_{2}	13153_{10}	468_{16}	9711_{10}	452_{8}	110000111000_{2}	$DEB3_{16}$	5171_{8}

Продолжение табл. 3.2

Вариант	Номер задания								
	№1	№2	№3	№4	№5	№6	№7	№8	№9
20	4052_{10}	1100101111_2	12680_{10}	707_{16}	9103_{10}	775_8	11000101110100_2	$577C_{16}$	3070_8
21	2915_{10}	11111001001_2	14307_{10}	$E20_{16}$	10973_{10}	677_8	1110110101000_2	$E58A_{16}$	7666_8
22	2669_{10}	1000111100_2	12296_{10}	$AE6_{16}$	10792_{10}	420_8	10111100100101_2	$BDDD_{16}$	7242_8
23	5771_{10}	100100111111_2	14435_{10}	$D3D_{16}$	7962_{10}	350_8	1011111101010_2	$89B8_{16}$	3717_8
24	2070_{10}	100101001111_2	12784_{10}	$E14_{16}$	8462_{10}	430_8	1101110111011_2	$E671_{16}$	1025_8
25	5617_{10}	1100000010_2	14539_{10}	$E66_{16}$	11186_{10}	212_8	1101010111010_2	$958F_{16}$	1255_8
26	3726_{10}	101111111101_2	11960_{10}	$2FA_{16}$	10278_{10}	335_8	11100111001010_2	$68D2_{16}$	2601_8
27	2044_{10}	11100001011_2	14820_{10}	$3AC_{16}$	10138_{10}	111_8	1011101110010_2	$B186_{16}$	4365_8
28	4292_{10}	11010101100_2	10398_{10}	$2AB_{16}$	9404_{10}	615_8	10110010110110_2	$2E43_{16}$	2464_8
29	4522_{10}	1001111111_2	11872_{10}	475_{16}	10341_{10}	756_8	10001100111101_2	$CD5D_{16}$	1476_8
30	3112_{10}	100001101111_2	11911_{10}	DDB_{16}	8705_{10}	517_8	1110110111111_2	$80A0_{16}$	1046_8
31	3582_{10}	1001011001_2	13898_{10}	$86B_{16}$	7584_{10}	464_8	1011110001110_2	$A818_{16}$	4522_8
32	3779_{10}	11000000001_2	12705_{10}	893_{16}	7958_{10}	346_8	11001101010101_2	$239D_{16}$	2421_8
33	4941_{10}	11001110011_2	11102_{10}	$DE5_{16}$	7390_{10}	537_8	1110001110011_2	$B1AB_{16}$	2653_8
34	2949_{10}	11010001010_2	13285_{10}	$3DD_{16}$	11355_{10}	277_8	1001111111101_2	$850A_{16}$	2214_8
35	2633_{10}	10000001101_2	14246_{10}	$9B8_{16}$	7924_{10}	760_8	10100110111011_2	$DDA2_{16}$	5113_8
36	5292_{10}	11110101000_2	13926_{10}	490_{16}	10735_{10}	751_8	1111001000101_2	$466B_{16}$	3012_8
37	3097_{10}	1110010010_2	13502_{10}	$FD3_{16}$	7350_{10}	620_8	1001101010110_2	$E979_{16}$	3221_8
38	5516_{10}	11110100110_2	11643_{10}	$C40_{16}$	10966_{10}	751_8	11100010011101_2	$ACDC_{16}$	7235_8
39	5335_{10}	1011111011_2	11731_{10}	AFA_{16}	11553_{10}	427_8	11111111001001_2	$F530_{16}$	5116_8

Продолжение табл. 3.2

Вариант	Номер задания								
	№1	№2	№3	№4	№5	№6	№7	№8	№9
40	2729_{10}	1010101000_2	10233_{10}	$8BF_{16}$	10472_{10}	772_8	11111101000001_2	$D6B9_{16}$	1015_8
41	3952_{10}	10010101101_2	12732_{10}	$C70_{16}$	7942_{10}	511_8	1100100000010_2	$2FFF_{16}$	1247_8
42	4523_{10}	1111010010_2	11899_{10}	600_{16}	10452_{10}	125_8	1111011110000_2	$6BD7_{16}$	1601_8
43	3354_{10}	10001101001_2	12708_{10}	$2E2_{16}$	8730_{10}	570_8	1010100110101_2	$F893_{16}$	6505_8
44	5431_{10}	10101101111_2	14439_{10}	$2D3_{16}$	11235_{10}	662_8	11011010110101_2	$87A1_{16}$	2404_8
45	2072_{10}	1111000110_2	13862_{10}	$E72_{16}$	8191_{10}	645_8	10110110100010_2	$8AF4_{16}$	2615_8
46	5858_{10}	10010101000_2	10885_{10}	$FD4_{16}$	9206_{10}	353_8	1000111011000_2	9368_{16}	7767_8
47	5575_{10}	10001000100_2	10821_{10}	$69E_{16}$	8425_{10}	623_8	10011100101101_2	4061_{16}	5413_8
48	4336_{10}	1010101000_2	15780_{10}	819_{16}	11576_{10}	472_8	10111010100011_2	$AF7F_{16}$	3522_8
49	4395_{10}	10101001110_2	12127_{10}	$C75_{16}$	10968_{10}	736_8	11011010110111_2	$A282_{16}$	1742_8
50	3396_{10}	11001001110_2	13344_{10}	BBA_{16}	7837_{10}	145_8	10111101011101_2	5336_{16}	3326_8
51	5063_{10}	1110010101_2	14001_{10}	$72C_{16}$	11329_{10}	240_8	11000111111010_2	$EC7B_{16}$	6002_8
52	4474_{10}	11010001001_2	11782_{10}	$CD3_{16}$	8499_{10}	354_8	11000110001010_2	$9F89_{16}$	4701_8
53	4944_{10}	1111011101_2	15459_{10}	$5C1_{16}$	10327_{10}	362_8	1101011100110_2	$62D8_{16}$	2333_8
54	4079_{10}	1000110010_2	13296_{10}	$F29_{16}$	8051_{10}	752_8	11011011110011_2	$BB80_{16}$	4565_8
55	2054_{10}	10111010100_2	11486_{10}	$98E_{16}$	7143_{10}	360_8	1011001000101_2	2449_{16}	5141_8
56	4062_{10}	10011001001_2	14089_{10}	$2F2_{16}$	7003_{10}	556_8	1110110110001_2	$C257_{16}$	3025_8
57	3747_{10}	1111101001_2	13753_{10}	$2BE_{16}$	9941_{10}	122_8	10101111011110_2	$8AAA_{16}$	1247_8
58	4832_{10}	1001111101_2	15380_{10}	269_{16}	9847_{10}	234_8	10011100101001_2	$DE1E_{16}$	3621_8
59	3491_{10}	11001010100_2	13369_{10}	732_{16}	8211_{10}	656_8	1111010100011_2	8007_{16}	6507_8

Продолжение табл. 3.2

Вариант	Номер задания								
	№1	№2	№3	№4	№5	№6	№7	№8	№9
60	4917_{10}	11100110110_2	15508_{10}	$3B3_{16}$	7090_{10}	745_8	1111010010101_2	$4E18_{16}$	4404_8
61	2754_{10}	10001010111_2	14783_{10}	$CF4_{16}$	7464_{10}	516_8	11110101110111_2	$E27B_{16}$	2636_8
62	2850_{10}	10001101101_2	14927_{10}	$8FD_{16}$	11896_{10}	771_8	1011111101010_2	$5A2F_{16}$	4262_8
63	5131_{10}	1000111100_2	12348_{10}	$8A1_{16}$	10861_{10}	244_8	1111001010110_2	$A8DC_{16}$	7146_8
64	3037_{10}	11100000110_2	15208_{10}	$F31_{16}$	7430_{10}	707_8	1011001101100_2	$66E6_{16}$	4635_8
65	4089_{10}	1110100110_2	10785_{10}	$EF7_{16}$	10241_{10}	505_8	10110000110101_2	3949_{16}	2063_8
66	5072_{10}	11110101001_2	12259_{10}	$4D2_{16}$	10528_{10}	325_8	10010001011101_2	$72A1_{16}$	4437_8
67	4036_{10}	10110010110_2	10706_{10}	$5BA_{16}$	8766_{10}	453_8	10110110000000_2	$576A_{16}$	7113_8
68	2423_{10}	11011110101_2	14286_{10}	$5FB_{16}$	9353_{10}	150_8	1111101111001_2	$F5E0_{16}$	5212_8
69	5627_{10}	1010100110_2	13093_{10}	$79F_{16}$	11959_{10}	715_8	11010101000101_2	$C94E_{16}$	3444_8
70	4559_{10}	1100001100_2	11490_{10}	801_{16}	8101_{10}	715_8	11001001100110_2	$21F6_{16}$	5076_8
71	3111_{10}	10010100111_2	10962_{10}	701_{16}	11939_{10}	411_8	1100111000010_2	$8BAF_{16}$	1572_8
72	4606_{10}	10111111101_2	13669_{10}	$9DB_{16}$	10217_{10}	202_8	1101011110000_2	$3DBD_{16}$	2601_8
73	4462_{10}	11100001011_2	11298_{10}	430_{16}	7900_{10}	303_8	1010100110101_2	$E110_{16}$	2446_8
74	2197_{10}	11010101100_2	11026_{10}	$DC1_{16}$	9896_{10}	155_8	1100101001011_2	5974_{16}	3612_8
75	2598_{10}	1001101101_2	13117_{10}	$9B8_{16}$	8007_{10}	204_8	1010010111010_2	$A27D_{16}$	6477_8
76	4777_{10}	1000110101_2	14951_{10}	EFB_{16}	10303_{10}	121_8	11100010011101_2	$A98B_{16}$	4406_8
77	4051_{10}	1010110010_2	15809_{10}	CEE_{16}	9095_{10}	240_8	11000100100101_2	$ADE2_{16}$	4525_8
78	5001_{10}	1000000010_2	14054_{10}	852_{16}	8274_{10}	270_8	1001010111000_2	$B18A_{16}$	4261_8
79	3351_{10}	1111001101_2	11967_{10}	$6B1_{16}$	10657_{10}	211_8	10001010010001_2	$2E43_{16}$	7066_8

85

86

Окончание табл. 3.2

Вариант	Номер задания								
	№1	№2	№3	№4	№5	№6	№7	№8	№9
80	2214_{10}	1001100110_2	12568_{10}	970_{16}	10644_{10}	107_8	1001010001000_2	$CD51_{16}$	5065_8
81	4905_{10}	1011000011_2	15837_{10}	435_{16}	11872_{10}	662_8	11000000101010_2	$C0B0_{16}$	3776_8
82	4008_{10}	1101000001_2	14747_{10}	$77C_{16}$	11905_{10}	150_8	11001110101001_2	$D918_{16}$	3326_8
83	4307_{10}	10110000010_2	10173_{10}	342_{16}	8561_{10}	146_8	10100011001001_2	5611_{16}	6202_8
84	4916_{10}	1101001101_2	13697_{10}	$7F5_{16}$	7566_{10}	164_8	10000001011101_2	$E52F_{16}$	4101_8
85	3025_{10}	1000011011_2	11991_{10}	$8F2_{16}$	11231_{10}	230_8	10110110001000_2	$70B2_{16}$	4333_8
86	5343_{10}	1000110000_2	10029_{10}	$CF7_{16}$	7119_{10}	307_8	1111100111011_2	$885D_{16}$	5707_8
87	3591_{10}	10111001100_2	14683_{10}	$D10_{16}$	8047_{10}	327_8	11010100000001_2	$E616_{16}$	7463_8
88	3821_{10}	10011101001_2	15132_{10}	$D3C_{16}$	11071_{10}	515_8	11000000100110_2	9424_{16}	5562_8
89	5758_{10}	11111101001_2	10044_{10}	$71C_{16}$	7172_{10}	220_8	1100001000100_2	$A787_{16}$	5712_8
90	5235_{10}	10000111111_2	13209_{10}	$A19_{16}$	8915_{10}	161_8	1101001110000_2	$B02F_{16}$	2712_8
91	3450_{10}	10000010001_2	10685_{10}	ABC_{16}	9115_{10}	704_8	1000000110001_2	$29E8_{16}$	5476_8
92	3066_{10}	1101001111_2	14058_{10}	355_{16}	11158_{10}	220_8	11001000110101_2	$CCF6_{16}$	3575_8
93	3194_{10}	11111110110_2	14402_{10}	558_{16}	8368_{10}	311_8	1110110100010_2	$FF45_{16}$	3727_8
94	3151_{10}	1001001110_2	12288_{10}	CBC_{16}	11572_{10}	675_8	1001110111001_2	$2BE0_{16}$	4371_8
95	2457_{10}	1101100011_2	12993_{10}	$F8B_{16}$	7246_{10}	522_8	1011001000010_2	$74A9_{16}$	6055_8
96	4022_{10}	10111110001_2	11695_{10}	$EF2_{16}$	8688_{10}	255_8	1010101111001_2	$37A7_{16}$	4764_8
97	4834_{10}	1110111001_2	14659_{10}	$DD7_{16}$	8207_{10}	577_8	1010110100010_2	$3A0A_{16}$	4106_8
98	4209_{10}	10110010001_2	11671_{10}	$E54_{16}$	8367_{10}	256_8	1111000100010_2	$43BE_{16}$	3540_8
99	3792_{10}	1001001000_2	15634_{10}	$9FB_{16}$	8314_{10}	326_8	1000010111000_2	$AC77_{16}$	6214_8

Таблица 3.3. **Контрольные задания. Практическая часть. Второй раздел**

Вариант	Задание
00	Написать программу вычисления произведения чисел, являющихся положительными. В программе организовать ввод 10 чисел с клавиатуры. Организовать вывод результата на экран.
01	Написать программу, которая заменяет в последовательности положительные числа на нули. Организовать ввод чисел последовательности с клавиатуры. Вывести результат на экран.
02	Написать программу, производящую поиск наибольшего делителя натурального числа **n**. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран. Если делителей нет, вывести единицу.
03	Написать программу, производящую поиск наибольшего общего делителя двух натуральных чисел **n** и **m**. В программе организовать ввод чисел **n** и **m** с клавиатуры. Организовать вывод результата на экран. Если делителей нет, вывести единицу.
04	Написать программу, производящую вычисление факториала натурального числа **n**. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран. *Примечание: факториал n! – это произведение всех натуральных чисел до **n** включительно, например, 5!=1*2*3*4*5.*
05	Написать программу, которая определяет, является ли натуральное число **n** степенью числа 2, т.е. 2^X=n, где x-натуральное число. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран.
06	Написать программу, которая определяет максимальное значение из 10 целых чисел. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
07	Написать программу, которая определяет среднее значение 10 целых чисел. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран. При выводе результата указать, является значение четным или нечетным.
08	Написать программу, которая определяет произведение двузначных элементов последовательности из 10 чисел, которые делятся на **n** нацело. В программе организовать ввод числа **n** и чисел последовательности с клавиатуры. Организовать вывод результата на экран.

Продолжение табл. 3.3

Вариант	Задание
09	Написать программу, которая выводит на экран все числа в диапазоне от 1 до 100, которые делятся на число **n** нацело. В программе организовать ввода числа **n** с клавиатуры. Если таких чисел нет, вывести число 0.
10	Написать программу, которая определяет количество положительных чисел из 10 введенных целых значений. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
11	Написать программу, производящую поиск из 10 целых чисел минимального среди положительных. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата.
12	Написать программу, которая возводит число **N** в степень **X** с использованием только операций сложения, вычитания, деления и умножения. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран. *Примечание: учесть, что $N^{-X}=1/N^X$.*
13	Написать программу, которая определяет сумму цифр числа **n**. В программе организовать ввод **n** с клавиатуры. Организовать вывод результата на экран. Если введенная цифра отрицательная, вывести 0. *Пример: число 123, сумма цифр=1+2+3=6*
14	Написать программу, которая определяет количество одинаковых чисел из 10 введенных целых значений. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
15	Написать программу, которая определяет количество букв «а» в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
16	Написать программу, которая с использованием операторов цикла определяет порядковый номер первой буквы «а» в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран. Если буквы «а» в строке **s** нет, вывести 0.
17	Написать программу, которая определяет общее количество букв в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран. Если букв в строке **s** нет, вывести 0.

Продолжение табл. 3.3

Вариант	Задание
18	Написать программу, которая определяет максимальную длину слова в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
19	Написать программу, которая определяет число слов в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
20	Написать программу, которая определяет количество чисел последовательности **A**, которые больше числа **n**. В программе организовать ввод числа **n** и чисел последовательности **A** с клавиатуры. Организовать вывод результата на экран.
21	Составить программу, которая выводит слово **s n** раз. В программе организовать ввод числа **n** и слова **s** с клавиатуры.
22	Составить программу, которая вычисляет значение функции y=ax+b при x=0, 1, 2, 3,...,10. Значения a и b вводятся с клавиатуры. Результат вычислений вывести в виде таблицы.
23	Даны монеты в 1,2 и 10 рублей и купюры по 100 и 1000 рублей. Составить программу, которая по введенной пользователем зарплате, вычислит минимальное количество купюр и монет каждого достоинства, которые следует выдать.
24	Напишите программу, которая запрашивает пароль. Если пользователь ввел пароль правильно, вывести «Доступ разрешен». Если пользователь пять раз ошибется, выйти из программы.
25	Составить программу, которая находит сумму первых чисел из последовательности **A**, произведение которых не превышает заданного числа **m.** Вывести результат на экран.
26	Составить программу, меняющую местами первую и последнюю цифру натурального числа **n**. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран.
27	Составить программу, определяющую количество пятерок, четверок и троек в зачетной книжке. В программе организовать ввод оценок с клавиатуры. Организовать вывод результата.
28	Написать программу, выводящую цифры словами. Например, ввод «123», вывод – «Один, два, три».
29	Составить программу, которая с использованием операции цикла копирует из строки **s** строку с позиции **n** до позиции **k**. В программе организовать ввод чисел **n, k** и слова **s** с клавиатуры.

Продолжение табл. 3.3

Вариант	Задание
30	Написать программу, которая производит определение и вывод элементов в наборе **A**, которые меньше своего левого соседа, и количество таких элементов. В программе организовать ввод чисел последовательности **A** с клавиатуры.
31	Написать программу для проверки, представляют ли элементы в наборе чисел **A** возрастающую последовательность. В программе организовать ввод чисел последовательности **A** с клавиатуры.
32	Написать программу, вычисляющую значение функции $y=3x+5$, если $x<2$; $y=2x^2$, если $x=2$; $y=2x^2+2x+1$, если $x>2$. Значения x вводятся с клавиатуры. Организовать вывод результатов. После каждого шага вычисления запрашивать у пользователя, следует ли завершить работу программы или ввести новое значение x.
33	Написать программу, которая определяет, сколько раз в последовательности **A** числа меняют знак. В программе организовать ввод чисел последовательности **A** с клавиатуры. Организовать вывод результата на экран.
34	Составить программу, вычисляющую среднюю зарплату работников предприятия. Если средняя зарплата по предприятию выше средней по стране, составляющей 30 тыс. рублей, вывести на экран соответствующее уведомление. В программе организовать ввод зарплат работников с клавиатуры.
35	Начальный вклад в банке равен 100 тыс.руб. Каждый месяц размер вклада увеличивается на **P** процентов от имеющейся суммы (с учетом повышений на предыдущих периодах). Написать программу, которая по **P** определяет, через сколько месяцев вклад увеличится более чем на 30 тыс.руб.
36	Составить программу, по которой компьютер находит произведение нечетных чисел, начиная с единицы, и до тех пор, пока на вопрос, задаваемый после каждого шага вычислений: «Продолжить вычисления? (Да/Нет)», отвечают «Да».
37	Горняки прокладывали туннель: в первый день они прошли n метров, а в каждый следующий день они проходили на 2% больше от сделанного в предыдущий. Составить программу, которая вычисляет, за сколько дней они прорубят туннель в m метров? Числа m и n вводятся с клавиатуры.

Продолжение табл. 3.3

Вариант	Задание
38	Составить программу проверки ученика на знание таблицы умножения от 2 до 9. По окончании опроса ставится оценка (50–74% – тройка, 75–94% – четверка, 95–100 % – пятерка)..
39	Написать программу, которая заменяет букву **w** на двоеточие в строке **s**. В программе организовать ввод строки **s** и буквы **w** с клавиатуры. Организовать вывод результата на экран.
40	Написать программу, которая определяет, является ли натуральное число **n** степенью числа 3, т.е. 3^X=n, где x – натуральное число. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран.
41	Написать программу, которая определяет среднее значение 10 целых чисел, являющихся положительными. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
42	Написать программу для простейшего шифрования слова. Буква должна заменяться ее номером в алфавитном порядке, из которого вычитается номер буквы в слове. Организовать ввод слова с клавиатуры. Зашифрованное слово вывести на экран.
43	Составить программу, которая вычисляет значение функции y=sin(x+b) при x=0, 1, 2, 3,...,10. Значение b вводятся с клавиатуры. Результат вычислений вывести в виде таблицы.
44	Составить программу, по которой компьютер предлагает отгадать «задуманное» им число.
45	Составить программу, по которой выводится список всех костей домино.
46	Составить программу, которая определяет, что ввел пользователь (число или текст). Результат вывести на экран.
47	Составить программу, которая определяет позицию первой буквы “a” в строке **s**. Отчет вести от самого правого символа. В программе организовать ввод строки **s** с клавиатуры.
48	Составить программу, определяющую подоходный налог (13 %) со всех работников предприятия. Если общие отчисления по предприятию выше 1 млн рублей, вывести на экран соответствующее уведомление. В программе организовать ввод зарплат работников с клавиатуры.

Продолжение табл. 3.3

Вариант	Задание
49	Составить программу, определяющую количество цифр в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
50	Написать программу, которая определяет сумму чисел, являющихся отрицательными. В программе организовать ввод 10 целых чисел с клавиатуры. Организовать вывод результата.
51	Написать программу, которая определяет среднее арифметическое, минимальный и максимальный элементы из 10 введенных чисел. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
52	Написать программу, производящую поиск наименьшего делителя натурального числа **n**. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран. Если делителей нет, вывести единицу.
53	Написать программу, производящую поиск наименьшего общего делителя двух натуральных чисел **n** и **m**. В программе организовать ввод чисел **n** и **m** с клавиатуры. Организовать вывод результата на экран.
54	Написать программу, которая определяет минимальное значение из 10 целых чисел. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
55	Написать программу, которая определяет сумму *двузначных* элементов последовательности из 10 чисел, которые делятся на **n** нацело. В программе организовать ввод числа **n** и чисел последовательности с клавиатуры. Организовать вывод результата.
56	Написать программу, которая выводит на экран все делители числа **n,** за исключением единицы и самого числа **n**. В программе организовать ввод **n** с клавиатуры. Если делителей нет, вывести **0**.
57	Написать программу для определения количества отрицательных чисел из 10 введенных целых значений. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата.
58	Написать программу, производящую поиск из 10 целых чисел максимального среди отрицательных значений. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.

Продолжение табл. 3.3

Вариант	Задание
59	Написать программу для определения количества разрядов числа **n**. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран. Если введенная цифра отрицательная, вывести 0.
60	Написать программу поиска суммы элементов массива, расположенных после минимального элемента. Организовать ввод элементов одномерного массива и вывод результата.
61	Написать программу, которая производит определение и вывод элементов в наборе **A**, которые больше своего правого соседа, и количество таких элементов. В программе организовать ввод чисел последовательности **A** с клавиатуры.
62	Написать программу, вычисляющую значение функции y=x+1, если x<0; y=x^2+2, если x≥0 и x<4; y=x^3+3, если x≥4. Значения x вводятся с клавиатуры. Организовать вывод результатов. После каждого шага вычисления запрашивать у пользователя, следует ли завершить работу программы или ввести новое значение x.
63	Составить программу, которая вычисляет, какое минимальное количество чисел надо взять, чтобы результат 1×2×3×4×5×..×n был четырехзначным числом. Вывести результат на экран.
64	Составить программу, которая вычисляет значение функции y=1/(x+b). Значения x и b вводятся с клавиатуры. Организовать вывод результатов. После каждого шага вычисления запрашивать у пользователя, следует ли завершить работу программы.
65	Составить программу, которая во введенном пользователем арифметическом выражении (введенной строке **s**) производит проверку соблюдения баланса открывающихся и закрывающихся скобок. При отсутствии баланса вывести уведомление.
66	Пара кроликов каждый год дает приплод (самку и самца), которые через 2 месяца способны давать новый приплод. Составить программу, вычисляющую, сколько кроликов будет через один, два и три года.
67	Составить программу вычисления квадрата суммы всех четных чисел от 2 до 100. Организовать вывод результата на экран.
68	Написать программу, выводящую трехзначное число словами. Например, ввод «123», вывод – «Одна тысяча, две сотни, три единицы». Ввод числа организовать с клавиатуры.

Продолжение табл. 3.3

Вариант	Задание
69	Составить программу, определяющую число знаков препинания в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
70	Дана последовательность **A** из 10 чисел. Написать программу определения количества чисел последовательности, которые меньше или равны числу **n**. В программе организовать ввод чисел **A** и **n** с клавиатуры. Организовать вывод результата.
71	Написать программу, определяющую количество букв «о» в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
72	Написать программу, определяющую порядковый номер последней буквы «а» в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран. Если буквы «а» в строке **s** нет, вывести 0.
73	Написать программу, определяющую число пробелов в строке **s**. В программе организовать ввод **s** с клавиатуры. Организовать вывод результата на экран. Если пробелов нет, вывести 0.
74	Написать программу, определяющую минимальную длину слова в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
75	Написать программу, которая определяет, является ли натуральное число **n** степенью числа 5, т.е. 5^X=n, где x-натуральное число. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран.
76	Написать программу, которая определяет среднее значение для чисел, являющихся отрицательными. В программе организовать ввод 10 чисел с клавиатуры. Организовать вывод результата.
77	Вычислить значения функции y=ax^2+bx+c при x=-10,-9,...,0. Значения a, b и c вводятся с клавиатуры. Результат вычислений вывести в виде таблицы.
78	Написать программу для простейшего шифрования текста. Буква должна заменяться на букву, которая следует за ней по алфавиту. Зашифрованное слово вывести на экран.
79	Написать программу, производящую поиск из 10 целых чисел минимального среди отрицательных. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата.

Продолжение табл. 3.3

Вариант	Задание
80	Написать программу, производящую вычисление частного от деления единицы на факториал натурального числа **n**. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран.
81	Написать программу, которая определяет количество элементов последовательности из 15 чисел, которые делятся на 2 и 3 нацело. В программе организовать ввод чисел последовательности с клавиатуры. Организовать вывод результата на экран.
82	Написать программу, которая выводит на экран все числа в диапазоне от 1 до 100, которые не делятся на число **n** нацело. В программе организовать ввод числа **n** с клавиатуры.
83	Написать программу, которая определяет количество нулей из 10 введенных значений. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
84	Составить программу, которая вычисляет сумму частных до n включительно, т.е. 1/1 + 1/2 + 1/3 + ... + 1/n. В программе организовать ввод числа **n** с клавиатуры. Организовать вывод результата на экран.
85	Написать программу, которая заменяет в последовательности отрицательные числа на нули. Организовать ввод чисел последовательности с клавиатуры. Вывести результат на экран.
86	Написать программу проверки принадлежности натурального числа **n** к простым. В программе организовать ввод числа **n** с клавиатуры. При положительном исходе вывести на экран «Число **n** – простое». В противном случае «НЕ простое».
87	Написать программу, которая определяет количество неповторяющихся чисел из 10 введенных целых значений. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.
88	Даны монеты в 1,2 и 5 рублей и купюры по 100 и 5000 рублей. Составить программу, которая по введенной пользователем зарплате, вычислит минимальное количество купюр и монет каждого достоинства, которые следует выдать.
89	Составить программу, которая вычисляет сумму всех целых чисел последовательности **A**. В программе организовать ввод чисел с клавиатуры. Организовать вывод результата на экран.

Окончание табл. 3.3

Вариант	Задание
90	Составить программу, определяющую число одинаковых соседних букв в строке **s**. В программе организовать ввод строки **s** с клавиатуры. Организовать вывод результата на экран.
91	Написать программу, выводящую значения элементов массива, которые расположены после максимального элемента. Организовать ввод элементов одномерного массива.
92	Составить программу проверки ученика на знание таблицы умножения. Опрос продолжается до тех пор, пока не будет получено 5 верных ответов подряд. Вывести оценку и количество заданных вопросов.
93	Составить программу, по которой компьютер находит сумму четных чисел, начиная с двойки, и до тех пор, пока на вопрос, задаваемый после каждого шага вычислений: «Продолжить вычисления? (Да/Нет)», отвечают «Да».
94	Составить программу, которая вычисляет, сколько слагаемых суммы 1+2+3+4+5+... надо взять, чтобы получить трехзначное число. Вывести результат на экран.
95	Составить программу, которая вычисляет значение функции y=ln(x+b) при x=1, 2, 3,...,10. Значение b вводится с клавиатуры. Результаты вычислений вывести на экран в виде таблицы.
96	В книге **N** страниц. Написать программу, которая определит, сколько цифр понадобится, чтобы пронумеровать все страницы книги. Вывести результат на экран.
97	Составить программу, вычисляющую процент работников предприятия, средняя зарплата которых выше средней по стране, составляющей 25 тыс. рублей. Вывести результат на экран. В программе организовать ввод зарплат работников с клавиатуры.
98	Написать программу, вычисляющую значение функции y=1, если x<-1; y=2x+2, если x≥-1 и x<0; $y=x^4+1$, если x≥0. Значения x вводятся с клавиатуры. Организовать вывод результатов. После каждого шага вычисления запрашивать у пользователя, следует ли завершить работу программы или ввести новое значение x.
99	Написать программу, которая определяет количество элементов последовательности из 20 чисел, которые делятся на 3 и 5 нацело. В программе организовать ввод чисел последовательности с клавиатуры. Организовать вывод результата на экран.

ЗАКЛЮЧЕНИЕ

В данном учебно-методическом пособии приведена программа дисциплины «Информатика». Указана трудоемкость изучаемой дисциплины, основные виды занятий, контролей. Определен объем аудиторной и самостоятельной работы. Приведены темы дисциплины, каждая из которых поддержана рекомендациями по изучению конкретных разделов рекомендуемой литературы. По всем темам представлен необходимый перечень вопросов для самоконтроля полноты изученного материала. Рекомендуемый объем самостоятельной работы достаточен для подготовки к сдаче экзамена по дисциплине.

Рассмотрены теоретические вопросы по темам «Системы счисления», «Основы алгоритмизации» и «Язык программирования Турбо Паскаль». Представлены примеры решения практических задач по каждому из заданий контрольной работы. Приведены рекомендации по выбору варианта контрольного задания и указания по выполнению работы. Представлены варианты заданий контрольной работы. Объем материала, изложенный в учебно-методическом пособии, необходим и достаточен для самостоятельного выполнения контрольной работы.

После самостоятельного изучения тем дисциплины и выполнения контрольной работы следует сформировать перечень сложных для понимания вопросов для их обсуждения во время аудиторных занятий.

ПРИЛОЖЕНИЯ

Приложение 1

Таблица П1.1. **Основные процедуры и функции языка Паскаль**

Имя и параметры	Тип	Действие
Read(a,b,...)	процедура	Вводит значения с клавиатуры в переменные a, b ...
Write(a,b,...)	процедура	Выводит значения a, b ... в окно вывода
Abs(x)	функция	Возвращает абсолютное значение (модуль) x
Sqr(x)	функция	Возвращает квадрат x
Sqrt(x)	функция	Возвращает квадратный корень из x
Sin(x)	функция	Возвращает синус x
Cos(x)	функция	Возвращает косинус x
Ln(x)	функция	Возвращает натуральный логарифм x
Exp(x)	функция	Возвращает e в степени x (e=2.718281...)
Arctan(x)	функция	Возвращает арктангенс x
Power(x,y)	функция	Возвращает x в степени y
Round(x)	функция	Возвращает результат округления x до ближайшего целого
Trunc(x)	функция	Возвращает целую часть x
Int(x)	функция	Возвращает целую часть x
Frac(x)	функция	Возвращает дробную часть x
Ord(x)	функция	Возвращает номер значения порядкового типа
Chr(x)	функция	Возвращает символ с кодом x
Odd(x)	функция	Возвращает True, если x - нечетное, и False в противном случае
Inc(x)	процедура	Увеличивает x на 1
Dec(x)	процедура	Уменьшает x на 1
Inc(x,n)	процедура	Увеличивает x на n
Dec(x,n)	процедура	Уменьшает x на n
Pred(x)	функция	Возвращает предыдущее значение порядкового типа
Succ(x)	функция	Возвращает следующее значение порядкового типа
Random(x)	функция	Возвращает случайное целое в диапазоне от 0 до x-1
Random	функция	Возвращает случайное вещественное в диапазоне (0..1)
Length(s)	функция	Возвращает длину строки
Concat(s1,s2,..)	функция	Возвращает строку, представляющую собой сцепление строк s1,s2,..
Copy(s,ind,cnt)	функция	Копирует из строки s cnt символов начиная с символа номер ind
Pos(subst,st)	функция	Отыскивает в строке st первое вхождение подстроки subst и возвращает номер позиции

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. **Информатика**. Базовый курс: учеб. пособие для вузов / под ред. С.В. Симоновича.– 2-е изд.- Спб: Питер, 2006. – 640 с.
2. **Могилев, А.В.** Информатика: учеб. пособие для вузов / А.В. Могилев, Н.И. Пак, Е.К. Хеннер; под ред. Е.К. Хеннера.– 7-е изд., стер.– М.: Академия, 2009. – 848 с.
3. **Фаронов, В.В.** Turbo Pascal 7.0. Начальный курс: учеб. пособие для вузов / В.В. Фаронов. – М.: КНОРУС, 2006.– 576 с.
4. **Фаронов, В.В.** Delphi. Программирование на языке высокого уровня: учеб. для вузов/ В.В. Фаронов.– Спб.: Питер, 2006. – 640 с.
5. **Малыхина, М.П.**. Базы данных: основы, проектирование, использование: учеб. пособие для вузов/ М.П. Малыхина. –2-е изд.– Спб.: БХВ-Петербург, 2007.– 528с.
6. **Карпов Б.И.** Самоучитель Visio 2003 / Б. И. Карпов.—СПб.: Питер, 2006.—335 с.
7. **Волков, В.Б.** Понятный самоучитель работы в Windows XP / В.Б. Волков.– СПб: Питер, 2006.– 202 с.
8. **Бэкон Д.** Операционные системы – Спб.: Питер, 2004.

СУЛЫНЕНКОВ Илья Николаевич,
КВАША Елена Николаевна

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
по дисциплине «Информатика» для студентов факультета заочного и вечернего обучения направления подготовки «Электроэнергетика и электротехника»

Редактор Т.В. Соловьева

Подписано в печать г. Формат 60х84 1/16.
Печать плоская. Усл. печ. л. 5,81. Уч.-изд. л. 4,31.
Тираж 50 экз. Заказ №
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»
Отпечатано в УИУНЛ ИГЭУ
153003 Иваново, ул. Рабфаковская, 34.