Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Тепловые электрические станции»

ИНФОРМАТИКА

Лабораторный практикум

Часть 1

Минск 2007

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Тепловые электрические станции»

ИНФОРМАТИКА

Лабораторный практикум
для студентов специальности
1-43 01 04 «Тепловые электрические станции»

В 2 частях

Часть 1

Минск 2007

УДК 004(076.5)
~~ББК 32.81 я 7~~
И 74

Составители:
Л.А. Тарасевич, Е.В. Пронкевич, Ю.Б. Попова

Рецензенты:
М.И. Фурсанов, В.А. Булат

И 74 Информатика: лабораторный практикум для студентов специальности 1-43 01 04 «Тепловые электрические станции». В 2 ч. Ч. 1 / Сост.: Л.А. Тарасевич, Е.В. Пронкевич, Ю.Б. Попова. – Минск: БНТУ, 2007. – 92 с.

Лабораторный практикум предназначен для обучения студентов работе на персональном компьютере, знакомит с основными понятиями операционных систем MS DOS и WINDOWS, инструментальной программой-оболочкой NORTON COMMANDER, текстовым редактором MS WORD, электронными таблицами MS EXCEL, основами программирования на языке TURBO PASCAL 7.0. В лабораторном практикуме приведено достаточное количество примеров решения задач и заданий для практического усвоения материала.

Лабораторный практикум предназначен для студентов специальности 1-43 01 04 «Тепловые электрические станции» дневной и заочной формы обучения.

ISBN 978-985-479-726-7 (Ч. 1)
ISBN 978-985-479-784-7

СОДЕРЖАНИЕ

Введение

Данный лабораторный практикум написан в рамках изучения курса информатики студентами технических специальностей. В лабораторном практикуме рассмотрены операционные системы MS DOS и WINDOWS, инструментальная программа-оболочка NORTON COMMANDER, текстовый редактор MS WORD, электронные таблицы MS EXCEL, язык программирования TURBO PASCAL 7.0. Знание изложенного материала позволяет приобрести практические навыки работы на компьютере.

Система программирования TURBO PASCAL, разработанная американской фирмой BORLAND, по-прежнему остается одной из самых распространенных систем. Этому способствует простота освоения языка, возможность создания структурированных программ для решения как вычислительных задач, так и задач, связанных с обработкой сложных структур данных. Появившиеся в настоящее время инструментальные средства для разработки программ, такие как BORLAND PASCAL, DELPHI, работающие в WINDOWS, основываются на TURBO PASCAL и развивают его идеи.

Лабораторная работа № 1

ОПЕРАЦИОННАЯ СИСТЕМА MS DOS

Цель работы: закрепление практических навыков работы с операционной системой **MS DOS**.

Теоретические сведения

Операционная система (ОС) – это главная программа, управляющая работой компьютера в целом. На персональных компьютерах типа IBM PC используются в основном операционные системы **MS DOS** и **WINDOWS.**

Операционная система **MS DOS** – это самая простая операционная система для компьютеров IBM PC. Она используется на всех младших моделях IBM PC и может применяться на всех старших моделях компьютеров этого же типа.

Операционная система **WINDOWS** – современная и удобная операционная система для старших моделей персональных компьютеров IBM PC.

Основными объектами во всех операционных системах на ПК являются файлы, программы и каталоги. Все программы в ПК представляются отдельными файлами или наборами файлов, хранящихся в определенном каталоге.

Файлы – это последовательность записей на машинных носителях – магнитных или оптических дисках, магнитных или перфолентах и т. п. Все данные и программы на ПК записываются в виде файлов или наборов файлов. Все файлы в памяти ПК имеют уникальные имена.

Совокупности файлов в памяти ПК объединяются в форме каталогов (или директориев, или папок) и подкаталогов. Каждый каталог имеет свое уникальное имя. Имя подкаталога образуется из его собственного имени и имени каталога, в котором он находится.

В операционных системах **MS DOS** и **WINDOWS имена файлов** образуются из латинских букв и цифр с добавлением трехбуквенных окончаний после точки. Для записи окончаний в этих операционных системах приняты правила:

.exe – исполняемый файл;
.com – исполняемый файл;
.bat – командный файл;
.txt – текстовый файл;
.doc – текстовый файл.

Работа с любыми операционными системами – это в основном работа с каталогами файлов и программ, размещенных на магнитных и оптических дисках. Эта работа состоит в просмотре каталогов и подкаталогов, копировании файлов и запуске тех или иных программ.

В **DOS** работой управляет командный процессор. Он находится в файле **COMMAND.COM**. Команды, которые командный процессор выполняет самостоятельно, называются *внутренними*. Для выполнения *внешних* команд процессор ищет нужную программу, загружает ее в память и передает ей управление.

Команда – это строка символов (состоит из имени команды или вызываемой программы), вводимая пользователем с клавиатуры на приглашение **DOS**.

Команды для работы с файлами и каталогами:

имя_диска: – переход на другой диск;

md [диск:\] имя_каталога – создание подкаталога;

cd [диск:\] путь – смена каталога;

dir [диск:\] [путь\][имя_файла] – просмотр каталога;

rd [диск:\] имя_каталога – удаление пустого каталога;

copy con имя_файла – создание небольшого текстового файла (после ввода этой команды нужно поочередно вводить строки файла. После ввода последней строки нажать клавишу **F6**, а затем **Enter**);

ren [диск:\ путь] имя_файла1 имя_файла2 – переименование файла;

copy [диск:\путь] имя_файла1 [диск:\путь] имя_файла2 – копирование файлов;

copy имя_файла1+имя_файла2 имя_файла3 – соединение файлов 1 и 2 в 3;

del [диск:\путь] имя_файла – удаление файла.

Поскольку в современных компьютерах используются операционные системы **WINDOWS** различных версий, то с командами **MS DOS** можно ознакомиться либо с загрузочного диска, либо из командной строки **DOS**. Вызвать командную строку **DOS** можно последовательностью действий стартового меню Пуск ▶ Программы ▶ Стандартные ▶ Командная строка. Для выхода из командной строки **DOS** существует команда **exit**.

Для упрощения работы с ПК используют различные программы-оболочки. До недавнего времени самой популярной из них была **NORTON COMMANDER (NC)**. Запуск **NORTON COMMANDER** осуществляется набором в командной строке **[диск:\][путь\] nc.** После запуска на экране появляются две прямоугольные панели. Ниже располагается командная строка **DOS**. Еще ниже строка с назначениями клавиш **NC**.

Управление панелями NC осуществляется нажатием клавиш (здесь плюс обозначает одновременное нажатие):

<TAB> – сделать активной другую панель;

<Alt>+<F1>, <Alt>+<F2> – выбор диска в левой/правой панели;

<Ctrl>+<O> – убрать/вывести панели на экран;

<Ctrl>+<P> – убрать/вывести неактивную панель на экран;

<Ctrl>+<F1>, <Ctrl>+<F2> – убрать/вывести левую/правую панель на экран;

<Ctrl>+<U> – поменять панели местами.

Операции над выделенными файлами и каталогами выполняются функциональными клавишами:

<F1> – Help – помощь в работе с **NC**;

<F2> – Menu – вызов меню пользователя;

<F3> – View – чтение (просмотр) файла;

<F4> – Edit – редактирование файла. Для создания нового файла надо нажать **<F4>+<Shift>;**

<F5> – Copy – копирование файла/каталога;

<F6> – RenMov – переименование/перенос файла/каталога;

<F7>– MkDir – создание подкаталога;

<F8> – Delete – удаление файла/каталога;

<F9> – PullDn – вызов управляющего меню **NC**;

<F10> – Quit – выход из **NC**.

Как было указано выше, с помощью клавиши **<F9>** можно вызвать управляющее меню. В верхней строке экрана появится строка с пунктами: **Left, Files, Disk, Commands, Right**.

Пункты меню **Left** и **Right** задают режимы вывода информации в левой/правой панелях. Подменю содержит следующие пункты:

Brief – отображается краткая информация о файлах;

Full – отображается полная информация о файлах;

Info – отображается информация о каталоге и диске на другой панели;

Tree – отображается дерево каталогов;

quick View – отображается содержимое файла;

Find file panel – отображаются найденные файлы;

Link – устанавливается/отменяется режим связи между компьютерами;

Name – файлы выводятся в алфавитном порядке;

Extension – файлы выводятся в алфавитном порядке расширений;

Time – вывод файлов в порядке убывания даты модификации;

Size – файлы выводятся в порядке убывания их размеров;

Unsorted – файлы выводятся в неотсортированном порядке;

Re-read – повторное чтение оглавления каталога;

Filter – режим отображения части файлов из каталога;

Drive – переход на другой диск.

Пункты меню **Files** дают возможность производить различные операции с файлами и дублируют основные команды функциональных клавиш.

Пункты меню **Disk** содержат сервисные команды для работы с дисками: создание копии дискеты, форматирование дискет, установка меток на диске, работа с сетью, очистка диска от ненужных файлов.

Пункты меню **Commands** служат для выполнения сервисных функций: вывода на экран дерева каталогов, поиск файла на диске, просмотр команд, введенных с командной строки, сравнение каталогов панелей, редактирование списка команд по нажатию пользователем клавиши **F2**.

Варианты заданий

Задание № 1

1. Перейти на рабочий диск, затем в рабочий каталог группы.
2. Создать каталог со своей фамилией.
3. В своем каталоге создать текстовый файл *file.txt*, заполнить его текстом. Вывести на экран содержимое файла.
4. В своем каталоге создать подкаталог со своим именем и скопировать туда созданный ранее текстовый файл.
5. В каталоге со своим именем создать подкаталоги *my 1*и *my2*.
6. Скопировать файл *file.txt* в каталог *my1* под именем *file1.txt*.
7. Скопировать файл *file.txt* в каталог *my2*. Переименовать его в *file2.txt*.

8. Соединить файлы *file1.txt* и *file2.txt* в *file.txt*.

9. Просмотреть содержимое своего каталога.

10. Удалить файлы из каталогов *my1* и *my2*, а затем и сами каталоги.

11. В каталоге со своей фамилией создать файл *report.txt*, в который записать изученные в ходе выполнения команды работы с файлами и каталогами.

12. Выйти из командой строки **DOS**.

Задание № 2

1. Запустить программу **NORTON COMMANDER (NC)**.

2. Выполнить пункты 1–7 и 9–11 из задания № 1 при помощи команд **NC**.

3. Ввести команду **DOS**: **time**. На запрос системы, если нужно, ввести новое время и нажать **Enter**.

4. Убрать панели, чтобы посмотреть результат выполнения команд. Отобразить панели.

5. Используя управляющее меню **NC**, изменить формат и порядок вывода файлов на панели: а) полный, сортировка по времени; б) полный, сортировка по размеру; в) краткий, сортировка по имени.

6. Вывести на панели информацию о диске.

7. Выйти из программы **NORTON COMMANDER**.

Контрольные вопросы

1. Архитектура современного компьютера.

2. Что такое операционная система? Типы ОС.

3. Назвать основные команды для работы с файлами и каталогами **MS DOS**.

4. Назвать основные клавиши управления панелями в **NC**.

5. Назвать основные клавиши для работы с каталогами и файлами в **NC**.

Лабораторная работа № 2

ОПЕРАЦИОННАЯ СИСТЕМА WINDOWS

Цель работы: закрепление практических навыков работы с операционной системой **WINDOWS**.

Теоретические сведения

WINDOWS – высокопроизводительная, универсальная, многозадачная, многопоточная операционная система с графическим пользовательским интерфейсом. После загрузки системы на экране отображается *Рабочий стол*. На нем располагаются значки, открывающие доступ ко всем ресурсам компьютера, локальной сети и Интернету.

Управление окнами

Запущенная программа в системе **WINDOWS** свою информацию отображает в отдельном окне. Одновременно можно запустить несколько программ. Неизбежно их окна будут перекрывать и/или покрывать друг друга (причем лишь *активное окно* будет иметь **синюю** строку заголовка, а остальные – **серую**; чтобы сделать нужное окно активным, требуется кликнуть мышью в любом его месте). При этом для удобства пользователя можно управлять размерами и положением каждого окна на рабочем столе. Тремя кнопками в правом углу строки заголовка окна можно: 1) закрыть окно (и тем окончательно снять его программу); 2) свернуть окно (и тем убрать его с рабочего стола, но поместить его *закладку* на *панель задач* – см. ниже пункт 3); 3) развернуть окно на весь экран либо восстановить его как часть рабочего стола. Для перемещения окна по рабочему столу требуется: поместить курсор мыши в заголовке; после чего, нажав, не отпуская, левую кнопку (говорят еще «зацепив»), перетащить мышью окно в нужное место стола; наконец отпустить кнопку мыши.

Панель задач

Внизу экрана на рабочем столе обычно находится *панель задач*, которую легко распознать по кнопке в левом углу. Если кликнуть

по кнопке **Пуск**, то появляется *стартовое меню* – ряд выпадающих окон со стрелками, по которым можно добраться до нужной *программы компьютера* и запустить ее (кликнув по названию).

На панели задач также могут быть расположены *закладки окон* с сокращенными их названиями. Если кликнуть мышью по закладке, то откроется соответствующее окно на рабочем столе.

В правом углу панели задач обычно отображается текущее *время* (*дату* можно узнать, если подвести к изображению времени курсор мыши). Рядом со временем находится окошко переключения регистра клавиатуры с русского (КРАСНЫЕ буквы) на английский (ЧЕРНЫЕ буквы) шрифт и наоборот. Переключиться можно, кликнув мышью по окошку, а затем – по названию нужного регистра. Можно переключиться на требуемый регистр и с клавиатуры, если нажать одновременно две клавиши: **<Alt>+<Shift>** или **<Ctrl>+<Shift>**.

Контекстное меню

Правая кнопка мыши используется для вызова контекстного меню, содержащего указания на операции, которые можно выполнить над выделенным объектом.

Программа Проводник

Вся информация хранится на дисковых устройствах в виде *файлов*, разложенных по *папкам*. Для навигации по файловой системе можно использовать специальную программу – **Проводник** системы **WINDOWS** (быстрый вызов программы **Проводник**: на кнопке Пуск в панели задач вызов контекстного меню, затем – раздел Проводник). В левом окне программы **Проводник** – структура файловой системы компьютера, в правом – содержимое выбранной папки. Ширину окон можно менять мышью. Перемещение вдоль по дереву устройств и папок – с помощью линеек прокрутки справа и внизу на рамке окна. Развернуть/свернуть дерево – клик мышью по значку +/–; показать содержимое папки в правом окне – клик по ярлыку папки. Двойной щелчок по *имени* исполняемого (с *расширением* .exe) файла приведет к запуску программы, хранящейся в этом файле.

Завершение работы в WINDOWS

Перед выключением компьютера или для передачи его другому пользователю следует: сохранить нужное в будущем содержимое окон в своей именной папке на диске, а затем закрыть все окна задач.

Варианты заданий

Задание № 1

1. Открыть окно программы **Мой компьютер**. Затем открыть окно системной папки **Корзина**. Варьируя размерами и положением на рабочем столе этих окон добиться, чтобы одно выглядывало из-под другого. Научиться переходить из окна в окно. С помощью линеек прокрутки научиться листать содержимое окон.

2. Загрузить программу **Блокнот** по цепочке пунктов стартового меню: Пуск ▶ Программы ▶ Стандартные ▶ Блокнот. Развернуть окно на весь экран. Занести в окно небольшой текст, содержащий русские и латинские буквы. Записать в тетрадь, как на вашем компьютере делается переключение регистров с клавиатуры.

3. Свернуть в закладку окно программы **Блокнот**. При показе результата преподавателю развернуть закладку на рабочем столе.

4. Курсор в тексте окна программы **Блокнот**; вызвать контекстное меню. Записать в тетрадь выделенные ярко пункты меню. Кликнуть по какому-нибудь выделенному пункту и описать в тетради результат этой операции.

5. Выделить курсором мыши часть текста (удерживая левую кнопку перемещать мышь, затем отпустить левую кнопку) и вызвать контекстное меню. Записать в тетрадь новые пункты меню.

6. Подсчитать количество дисковых устройств вашего компьютера, выполняя навигацию в программе **Проводник**. Просмотреть содержимое диска D.

7. Завершить работу в **WINDOWS**.

Задание № 2

1. Выполнить пункты 1–7 и 9–11 из задания № 1 лабораторной работы № 1 при помощи проводника **WINDOWS**.

2. Открыть файл *report.txt*, затем закрыть его. Узнать его размер и размер своей папки.

3. Создать ярлык для своей папки и переместить его на *Рабочий стол*. Удалить ярлык.

4. Выполнить в справочной системе **WINDOWS** поиск по теме «Поиск файлов». Записать необходимую информацию в файл *report.txt*.

5. Найти на рабочем диске все файлы и папки:

– содержащие в названии слово *report*;
– содержащие в тексте вашу фамилию;
– созданные или измененные на прошлой неделе;
– типа *Точечный рисунок*;
– типа *Текстовый документ*;
– типа *Электронные таблицы*.

Контрольные вопросы

1. Как вызвать панель задач?
2. Как вызвать контекстное меню?
3. Как создать папку, текстовый файл?
4. Как удалить папку, текстовый файл?
5. Как запустить основные программы **MS OFFICE**?

Лабораторная работа № 3

ТЕКСТОВЫЙ РЕДАКТОР MS WORD

Цель работы: закрепление практических навыков работы с текстовым редактором **MS WORD**.

Теоретические сведения

Текстовый редактор **MS WORD** является мощным текстовым процессором с огромным перечнем функционала. Несмотря на это, **MS WORD** обладает средствами (панели инструментов и контекстное меню), которые делают его простым в использовании. Используя раскрывающиеся меню и разнообразные окна можно быстро получить доступ к сотням функций. Некоторые возможности редактора **MS WORD** рассмотрим ниже.

WORD позволяет быстро создавать нумерованные и маркированные списки, облегчающие восприятие текста. Для этого можно использовать команду меню ***Формат – Список***, выбрав наиболее подходящую вкладку ***Маркированный***, ***Нумерованный*** или ***Многоуровневый список*** (рис. 3.1).

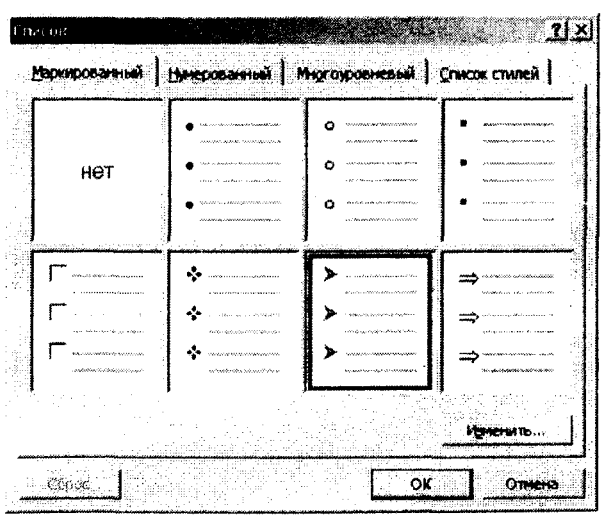


Рис. 3.1. Создание маркированного списка

Создание многоуровневого нумерованного списка:

1. В меню ***Формат*** выберите команду ***Список***, а затем – вкладку ***Многоуровневый***.

2. Выделите необходимый формат списка, а затем нажмите кнопку ***OK*** (рис. 3.2).

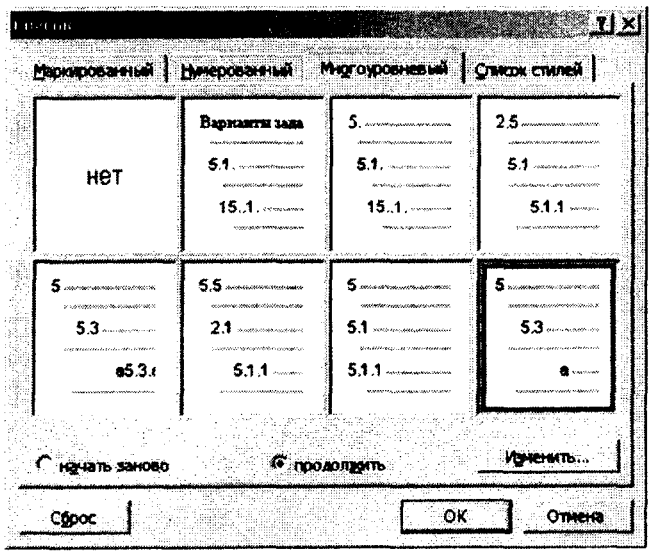


Рис. 3.2. Создание многоуровневого списка

Введите текст, нажимая клавишу ***ENTER*** после каждого элемента списка.

При помощи кнопок  ***Увеличить отступ абзаца*** и  ***Уменьшить отступ абзаца*** на панели инструментов ***Форматирование*** или клавиш Tab и Shift + Tab на клавиатуре расставьте уровни иерархии (подпункты).

Для более детальной настройки внешнего вида списка служит диалоговое окно ***Изменение многоуровневого списка*** (рис. 3.3).

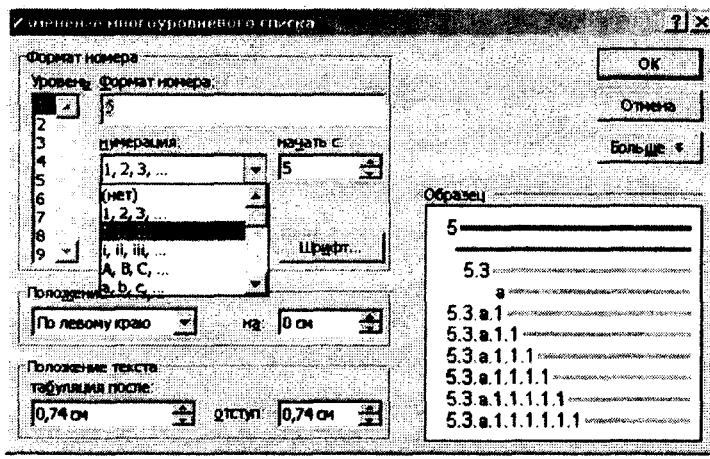


Рис. 3.3. Изменение многоуровневого списка

WORD позволяет вставлять фигурный текст с помощью коллекции **WordArt**. Для вставки ***объектов WordArt*** можно использовать команду меню ***Вставка – Рисунок – объект WordArt.*** При помощи панели инструментов ***WordArt*** следует выбрать готовый образец написания текста из коллекции, изменить формат шрифта, цвет, размер, обтекание текстом, форму надписи, выравнивание и прочие атрибуты фигурного текста.

Чтобы вставить в документ объект с экрана (пиктограмму программы, таблицу, открытое окно или его часть), следует:

a) скопировать в буфер обмена изображение с помощью комбинаций клавиш ***Alt + Print Screen*** или кнопки ***Print Screen***. Комбинация клавиш ***Alt + Print Screen*** позволяет скопировать в ***буфер обмена*** активное окно (окно, в котором выполняются операции, например, окно **WORD** или **Рабочий стол**). Кнопка на клавиатуре ***Print Screen*** позволяет скопировать весь экран;

b) вставить скопированное изображение в документ **WORD** командой ***Правка –Вставить***;

c) щелчком мыши выделить вставленный объект;

d) используя кнопку ***Обрезка*** панели инструментов ***Настройка изображения*** (рис. 3.4), передвигая мышью черные маркеры выделения, обрезать объект до нужных размеров.

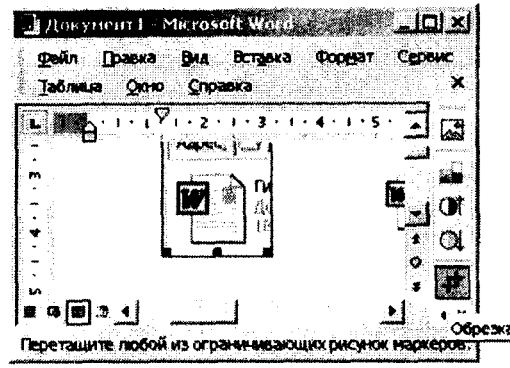


Рис. 3.4. Обрезка объекта

Для вставки объекта – электронной таблицы, фотографии и т.д. используйте команду ***Вставка – Объект***. Обычно такой объект вставляется в виде значка. Можно создать новый объект прямо в **WORD** или открыть уже существующий файл с помощью кнопки ***Обзор*** на вкладке ***Создание из файла*** (см. рис. 3.5). Для работы с этим объектом **WORD** вызовет соответствующее приложение (например, приложение **MS EXCEL**). Разумеется, эти приложения должны быть зарегистрированы в **WINDOWS**.

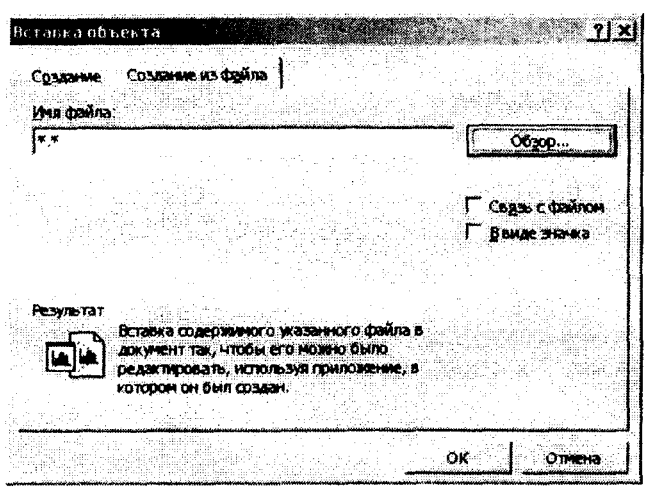


Рис. 3.5. Вставка объекта

Для создания таблицы следует выбрать команду меню ***Таблица – Вставить – Таблица.***

Таблица появляется на экране в виде прямоугольника, который расчерчен на строки и столбцы, в результате чего образуются ячейки.

Для определения координат ячейки необходимо сначала определиться с номером строки – 1, 2, 3 и т. д., а затем в этой строке отсчитать нужный столбец – A, B, C и т. д. (рис. 3.6).

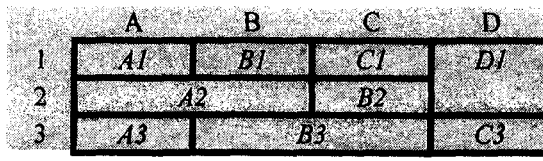


Рис. 3.6. Определение координат ячеек таблицы

Объединение и разбиение ячеек, удаление строк, столбцов и отдельных ячеек осуществляются при помощи соответствующих команд меню ***Таблица*** или кнопок панели инструментов ***Таблицы и границы.***

Вычисление – это процесс обработки данных и отображения результатов в ячейках таблицы.

Выполнение вычислений в таблице:

1. Выделите ячейку, в которую будет помещен результат.
2. Выберите команду меню ***Таблица – Формула.***
3. Любая формула начинается со знака = . При вычислении можно воспользоваться арифметическими операторами: +, –, *, /, ^, %.

Например: ***=A1+B1***, где A1 и B1 – координаты ячеек таблицы.

4. В списке ***Вставить функцию*** можно выбрать функцию, в скобках, через точку с запятой или двоеточие, укажите координаты ячеек. Например, для суммирования чисел из диапазона A1:B3 применяют формулу: = ***SUM(A1:B3).***

Наиболее часто используемые функции: ***SUM*** – сумма; ***PRODUCT*** – произведение; ***AVERAGE*** – среднее значение; ***MIN*** – минимальное значение; ***MAX*** – максимальное значение. Для ссылки на диапазон ячеек допустимы значения: ***ABOVE*** – выше; ***BELOW*** – ниже; ***LEFT*** – слева; ***RIGHT*** – справа.

Примеры формул в таблице:

{=MAX(A1;B1;C1)} – наибольшее значение из ячеек A1, B1 и C1.

{=A1*20%} – 20 % от числа в ячейке A1.

{=SUM(ABOVE)} – сумма ячеек, расположенных в столбце выше этой формулы (до первой пустой).

5. В поле ***Формат числа*** можно выбрать формат для чисел. Например, для отображения чисел в виде процентов следует выбрать: **0,00%.**

Результат вычисления будет помещен в виде поля в ту ячейку, где стоял курсор.

Варианты заданий

Задание № 1

Требуется:

1. Создать маркированный список с фамилиями подгруппы.

2. Создать многоуровневый список изучаемых в текущем семестре дисциплин с указанием разделов и тем.

3. Вставить объект **WordArt** со следующими параметрами:

a) текст надписи – своя фамилия и инициалы;

b) стиль объекта выбрать по своему усмотрению;

c) положение – перед текстом, можно применить поворот.

4. Вставить любой рисунок из коллекции рисунков **MICROSOFT OFFICE**.

5. Придумать и вставить название для рисунка.

6. Лист с выполненным заданием № 1 представить в альбомной ориентации (рис. 3.7).

Пример выполнения

- Иванов
- Петров
- Титарчук
- Крабов

Вариант: И. И.

1) Математика
   a) тригонометрия
   b) ряды
   c) интегралы
      - определенные
      - неопределенные
2) Физика
   a) механика
      - динамика
      - статистика
   b) термодинамика
   c) электричество
   d) оптика
3) Химия
   a) органическая
   b) неорганическая
4) Литература
   a) русская
   b) белорусская
5) Иностранный

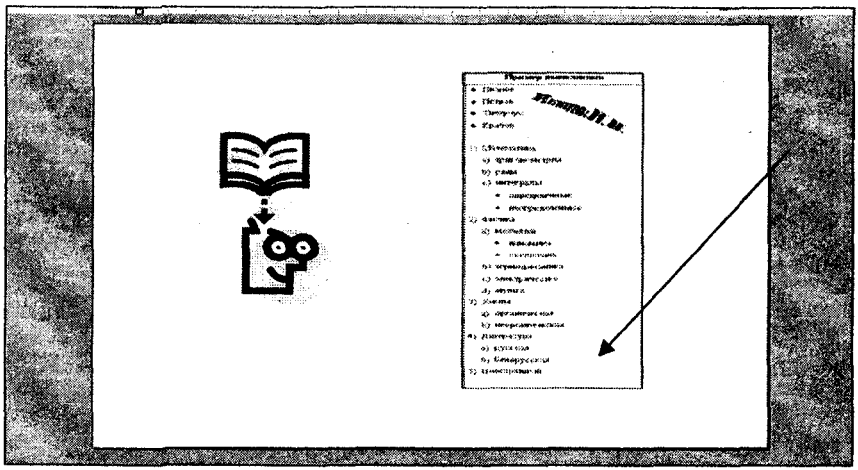


Рис. 3.7

Задание № 2

Требуется:

1. Создать, отформатировать и заполнить исходными данными таблицу согласно варианту.

2. В таблицу с исходными данными ввести формулы вычислений и получить результаты.

3. Сделать копию таблицы.

4. Проиллюстрировать результаты вычислений своей таблицы с кодами полей и значениями полей, копируя экран и обрезая вставленный объект в документ **WORD** до нужного размера.

5. В копии таблицы изменить 4–5 значений исходных данных и обновить поля.

Варианты задания № 2

Вариант 1. Температура воздуха в городах мира.

Температура воздуха в городах мира			
Дата	Лондон	Рим	Минск
15.11.2003	10	15	0
16.11.2003	12	13	-3
17.11.2003	-	10	2
18.11.2003	8	14	4
19.11.2003	4	-	2
20.11.2003	-2	4	-2
Минимальное			
Максимальное			

Вариант 2. Таблица для расчета пени.

Пеня = 2 % от суммы за один день просрочки.

Расчет пени		
Сумма (руб.)	Кол – во дней (просрочки)	Пеня
100000	10	
500000	15	
643000	30	
90500	1	
25000	60	

Вариант 3. Таблица распределения затрат предприятия.
Удельный вес = (Сумма/Итого суммы)*100 %.

ВЕДОМОСТЬ распределения затрат предприятия		
Статьи затрат	Сумма	Удельный вес
Сырье и материалы	1864500	
Зарплата	725500	
Отчисления на социальное страхование	8300	
Амортизационные отчисления	82500	
Прочие расходы	952000	
ИТОГО		

Контрольные вопросы

1. Как создать нумерованные и маркированные списки?
2. Как вставить рисунок из коллекции рисунков **MICROSOFT OFFICE**?
3. Как создать таблицу и изменить размеры в ней?
4. Как выполнить вычисления с использованием функций (**SUM, PRODUCT, AVERAGE, MIN**)?
5. Как создать рисунок?

Лабораторная работа № 4

ЭЛЕКТРОННЫЕ ТАБЛИЦЫ MS EXCEL

Цель работы: закрепление практических навыков работы с электронными таблицами **MS EXCEL**.

Теоретические сведения

Электронные таблицы **MS EXCEL** предназначены для обработки данных и позволяют:

1. Проводить инженерные, финансово-экономические и другие прикладные вычисления с использованием встроенных функций и формул.
2. Строить графики и диаграммы.
3. Получать выборки по различным критериям, подводить итоги по группам данных.
4. Автоматизировать работу пользователя и расширять стандартные возможности путем использования языка программирования Visual Basic for Application. Рабочий лист **EXCEL** и его элементы представлены на рис. 4.1.

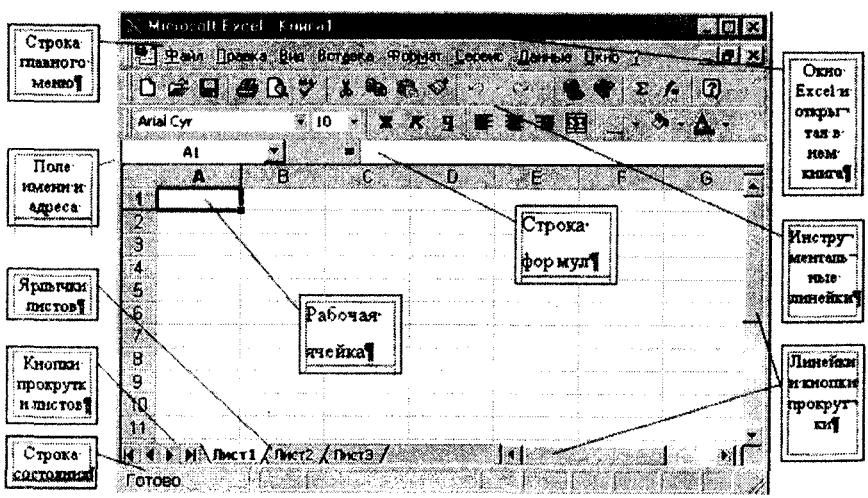


Рис. 4.1. Элементы рабочего листа

Ссылка на ячейку (**ее адрес**) однозначно определяется номером столбца и номером строки, например: A1, C5, D23. Ссылка на

непрерывный диапазон ячеек определятся адресами верхней левой и нижней правой ячеек, которые записываются через двоеточие, например: A2:C5, D12:G17, L2:L8. Ссылку в формуле можно ввести с клавиатуры, но чтобы избежать опечаток, лучше непосредственно щелкнуть левой клавишей мыши по нужной ячейке или выделить диапазон, тогда адрес в формуле будет прописан автоматически.

Ввод символов в **EXCEL** воспринимается как текстовая информация, цифр – как числовая. Для ввода чисел как текста набор начинают с символа кавычки ‘ , например, ‘222 – это является текстовой переменной. Ввод формул и функций начинается со знака = , за которым следуют адреса ячеек или переменных и знаки операций или функции, например, =A1+B1 суммирует значения, находящиеся в ячейках A1 и B1, или =СУММ(A1:A10) суммирует значения, находящиеся в интервале ячеек с адресами от A1 до A10.

В **EXCEL** для наглядного представления информации, содержащейся в таблицах, и ее анализа используются диаграммы. С помощью **Мастера диаграмм** можно строить различные типы диаграмм.

Структура диаграммы. Диаграмма всегда занимает прямоугольную область (рис. 4.2). В этой области размещены элементы определенных типов, которые все вместе формируют наглядное графическое представление числовых данных.

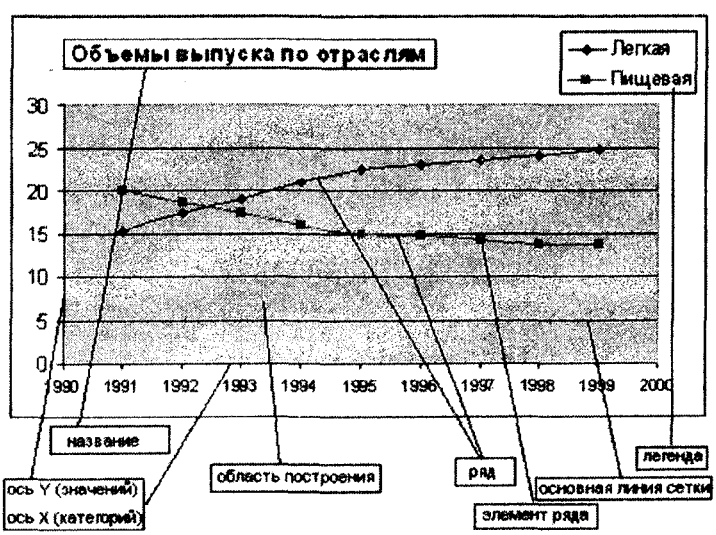


Рис. 4.2. Элементы диаграммы

Для создания внедренной диаграммы следует:

- выделить данные, которые будут использоваться в диаграмме. При построении диаграмм нажмите клавишу **CTRL** для выделения несмежных областей данных;
- щелкнуть по кнопке  **Мастера Диаграмм.**

В первом окне мастера пользователь определяет **тип** будущей диаграммы (рис. 4.3). Слева находится список основных типов, справа – подтипы для каждого из них.

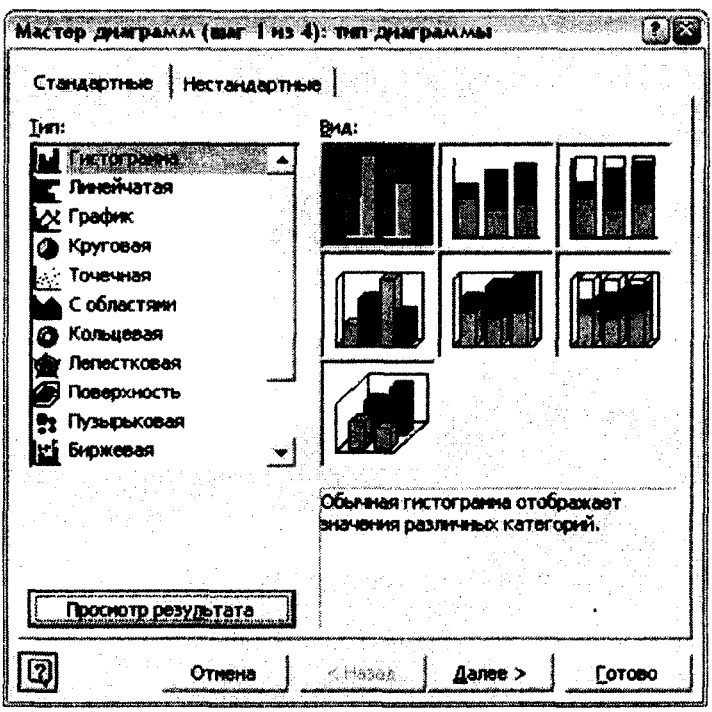


Рис. 4.3. Выбор типа диаграммы

Во **втором** диалоговом окне задаются **исходные данные** для построения диаграммы. Если перед вызовом мастера диаграмм пользователь выделил диапазон ячеек (рис. 4.4), то этот диапазон и будет первоначально считаться областью исходных данных диаграммы. Ряды могут располагаться в строках или столбцах.

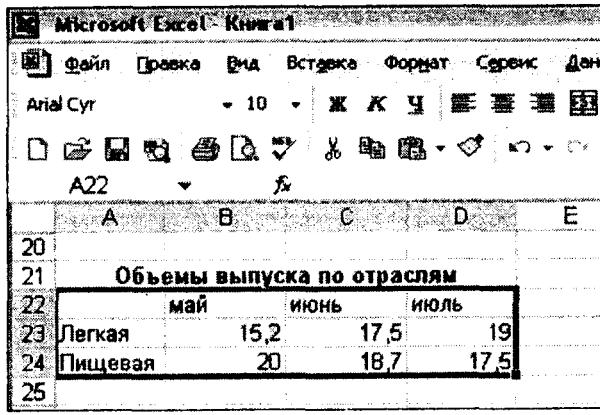


Рис. 4.4. Выделение исходных данных для построения диаграммы

Зная общий диапазон данных, **MS EXCEL** распределяет его по отдельным рядам. Если первый из рядов данных в таблице имеет тип, отличный от числового, то он автоматически интерпретируется как массив подписей по оси категорий. Если первые ячейки рядов содержат нечисловые значения, то они будут служить названиями рядов. При изменении параметров на диалоговом окне сразу отображается предварительный вид будущей диаграммы (рис. 4.5).

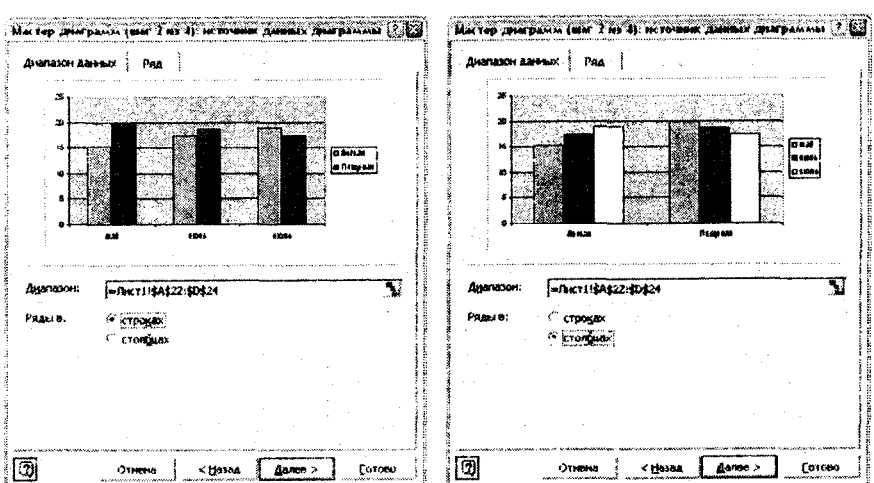


Рис. 4.5. Примеры интерпретации рядов данных в строках или столбцах

В третьем диалоговом окне задаются **параметры диаграммы.** Каждая вкладка отвечает за элементы диаграммы одного из перечисленных выше типов. Кроме них на диаграмму можно добавить **подписи данных** – значения элементов ряда или названия категорий около каждого элемента ряда, и **таблицу данных** – примыкающую к диаграмме таблицу, в которой собраны все исходные данные.

В четвертом диалоговом окне мастера задается **размещение диаграммы** – будет ли она занимать отдельный лист (тогда вводится имя этого листа) или она будет находиться на рабочем листе (рис. 4.6).

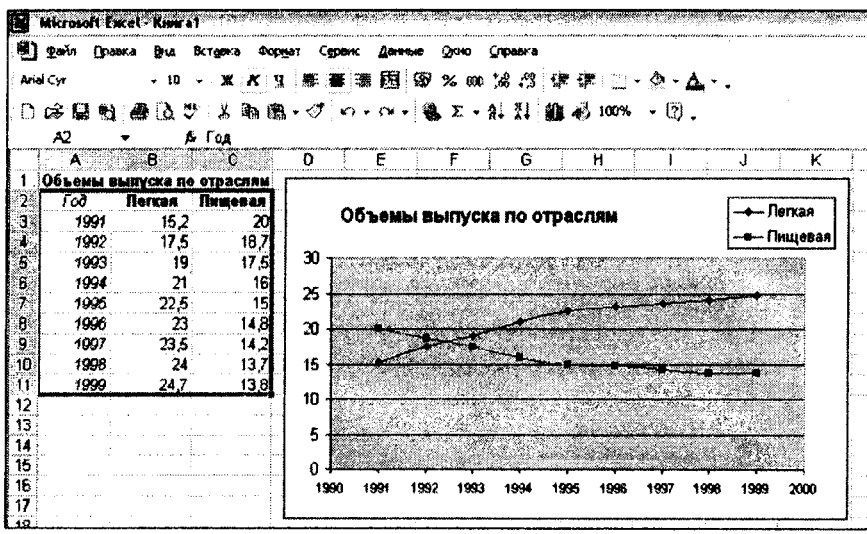


Рис. 4.6. Пример построения графика
по данным объемов выпуска продукции по отраслям

Варианты заданий

Вариант 1.

По заданным в таблице данным построить таблицу «ТЕМПЕРАТУРА ВОЗДУХА В ГОРОДАХ МИРА» (см. задание № 2 из лабораторной работы № 3 для этого варианта), выполнить расчеты и отобразить графически в виде лепестковой диаграммы температурные показатели городов мира.

При расчетах используйте функции:

МАКС, МИН, СРЗНАЧ

Вариант 2.

По заданным в таблице данным построить таблицу «РАСЧЕТ ПЕНИ» (см. задание № 2 из лабораторной работы № 3 для этого варианта), выполнить расчеты и отобразить графически в виде кольцевой диаграммы величину пени.

Вариант 3.

По заданным в таблице данным построить таблицу «ВЕДОМОСТЬ РАСПРЕДЕЛЕНИЯ ЗАТРАТ ПРЕДПРИЯТИЯ» (см. задание № 2 из лабораторной работы № 3 для этого варианта), выполнить расчеты и отобразить графически в виде линейной диаграммы удельные веса статей затрат предприятия.

Контрольные вопросы

1. Как построить таблицу в **MS EXCEL**?
2. Как отсортировать элементы в ячейках по возрастанию (убыванию)?
3. Как осуществляется ввод формул и функций в ячейках?
4. Как выполнить вычисления с использованием математических функций (**СУММ**, **МАКС**, **МИН**, **СРЗНАЧ**, **SIN** и т. д.)?
5. Как построить графики и диаграммы?

Лабораторная работа № 5

ВЫЧИСЛЕНИЕ ФУНКЦИИ

Выполнить лабораторную работу 2 способами:
1) с использованием типа переменных **Real;**
2) с использованием типа переменных **Extended.**

Цель работы: приобретение практических навыков составления линейных программ на языке **PASCAL** и представление алгоритма решения задачи в виде блок-схемы.

Теоретические сведения

Краткое описание среды TURBO PASCAL

Главное меню

Управление работой и настройка в системе **TURBO PASCAL** осуществляются с помощью команд, содержащихся в главном меню.

Поле главного меню содержит следующие команды:

FILE (файл), **EDIT** (редактирование), **SEARCH** (поиск), **RUN** (выполнение), **COMPILE** (компилирование), **DEBUG** (отладка), **TOOLS** (инструментарий), **OPTIONS** (опции), **WINDOW** (работа с окнами), **HELP** (помощь).

Команда **FILE**

Команда **FILE** содержит функции, управляющие работой с файлами. Активизация этой команды приводит к появлению на экране подменю, включающего следующие опции.

New – открытие нового файла с именем NONAMEXX.pas.

Open **([F3])** – загрузка файла с диска в новое окно редактора. При этом в диалоговом окне на выбор предлагаются имена файлов с расширением *.pas. Если ввести имя, которого нет в текущем каталоге, то будет создан новый файл.

Save (**[F2]**) – сохранение на диске текущего редактируемого файла. Если имя файла было NONAMEXX.pas, то среда запросит новое имя.

Save as – записывает содержимое окна на диск под другим именем.

Save all – сохраняет содержимое всех окон редактора в виде файлов на диске.

Change dir – изменяет текущий каталог пользователя.

Print – печатает содержимое текущего окна на принтере.

Printer setup – позволяет осуществить настройки принтера и параметров печати.

Dos shell – временный выход в **DOS**. Для возврата в среду **TURBO PASCAL** необходимо набрать команду ***Exit*** и нажать **[Enter]**.

Exit (**[Alt+X]**) – завершает работу **TURBO PASCAL**.

Команда **EDIT**

Undo (**[Alt+BackSpace]**) – отмена предыдущего выполненного действия в редакторе.

Redo – отменяет действие команды ***Undo***.

Cut (**[Shift+Del]**) («вырезать») – удаляет выделенный блок текста из окна редактора и переносит его в буфер обмена **Clipboard**.

Copy (**[Ctrl+Ins]**) – копирует выделенный блок из окна редактора в буфер обмена **Clipboard**.

Paste (**[Shift+Ins]**) («вставить») – копирование содержимого буфера обмена **Clipboard** в окно редактора, в место, где в данный момент находится курсор.

Clear (**[Ctrl+Del]**) – удаляет из окна выделенный блок без помещения его в буфер обмена.

Show clipboard – отображает содержимое буфера обмена.

Команда **SEARCH**

Find – поиск нужного фрагмента текста. В диалоговом окне при этом необходимо ввести ключевую строку для поиска.

Replase – отыскав заданный фрагмент текста, среда **TURBO PASCAL** заменяет его на новый фрагмент, который предварительно вводится в соответствующее диалоговое окно.

Search again – повторение поиска.

Go to line number – переход на строку редактора с указанным номером.

Команда **RUN**

Run (**[Ctrl+F9]**) – запускает компиляцию (трансляцию), компоновку (сборку, линковку) и выполнение программы без промежуточного участия программиста.

Go to cursor (**[F4]**) – на режиме отладки осуществляет прогон от выделенной голубым цветом строки до строки, в которой находится курсор.

Trace Into (**[F7]**) – запускает режим пошагового (построчного) выполнения программы (**трассировка**).

Step over (**[F8]**) – выполняет те же действия, что и **[F7]**, но без трассировки процедур и функций.

Program reset (**[Ctrl+F2]**) – прерывание режима отладки (трассировки). При этом гаснет подсветка текущей строки.

Команда **COMPILE**

Compile (**[Alt+F9]**) – компилирует ту программу, которая находится в окне редактора.

Make (**[Ctrl+F9]**) – компилирует программу вместе со всеми вызываемыми модулями пользователя, в которые были внесены изменения.

Build (**[Ctrl+F9]**) – в отличие от команды ***Make*** перекомпилирует все заказываемые модули вне зависимости от того, вносились ли в них изменения.

Команда **DEBUG**

Команда **DEBUG** предоставляет возможность проводить эффективную отладку программы, определять и изменять значения переменных в ходе трассировки, просматривать содержимое стека обращений к процедурам и т. д.

Команда **OPTIONS**

Команда **OPTIONS** обеспечивает управление параметрами интегрированной среды **TURBO PASCAL**, например, такими как настройки рабочих каталогов (директорий), конфигурация отдельных элементов и т. д.

Tile **(«черепица»)** – располагает окна так, чтобы каждое окно было видно на экране и все они имели бы равные размеры.

Cascade – окна располагаются каскадом, с перекрытием.

Zoom **([F5])** – раскрывает окно на весь экран или возвращает ему прежний вид.

Next **([F6])** – переключение от одного активного окна к другому.

Close **([Alt+F3])** – закрывает активное окно.

Основные понятия языка PASCAL

Языки программирования, как и любые другие языки, имеют свой алфавит. Алфавит языка программирования **PASCAL** состоит из символов трех видов:

1. Прописные и строчные буквы латинского алфавита: A, B, C, D, T, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, причем прописная и строчная буквы считаются одним символом.
2. Арабские цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
3. Специальные символы: (,), [,], {, }, ' (апостроф), . (точка), , (запятая), : (двоеточие), ; (точка с запятой), + (плюс), – (минус), * (звездочка), / (правый слеш), = (равно), > (больше), < (меньше), _ (знак подчеркивания), пробел (не имеет начертания).

Из символов алфавита составляются слова. Слова бывают двух типов:

– **зарезервированные (служебные)** – это слова, которые придумали разработчики языка программирования, вложили в них определенный смысл, который остается постоянным для всех программ языка **PASCAL** (например: **Begin**, **While**, **For**, **Repeat**);

– **идентификаторы (имена)** – это слова, которые придумывает программист; смысл этих слов сохраняется только в рамках одного алгоритма. Идентификатор может состоять из букв и цифр, но на первом месте должна стоять буква. Также в именах можно использовать знак подчеркивания. Другие специальные символы в именах использовать нельзя. Количество символов в имени не должно превышать 126. Однако пользоваться очень длинными именами неудобно, поэтому на практике используют короткие имена.

Примеры:

а) 32a – ошибка в том, что идентификатор начинается с цифры;

б) AB C – ошибка в том, что идентификатор содержит запрещенный знак – пробел;

в) Begin – это служебное слово, а не идентификатор;

г) Beta1 – правильная запись идентификатора;

д) Тэма – ошибка в том, что идентификатор содержит русскую букву «э»;

е) a_2_C – правильная запись идентификатора;

ж) РоМа – нельзя судить о правильности записи данного идентификатора, так как по внешнему виду нельзя определить состоит ли данный идентификатор только из латинских букв или там есть и русские буквы.

При составлении программы программист может использовать разные величины.

Константа – это величина, которая никогда не изменяется.

Текстовая константа может состоять из *любых* символов, которые есть на клавиатуре компьютера, независимо от того, входит этот символ в алфавит языка программирования Pascal или нет, и последовательность этих символов должна *обязательно* заключаться в *апострофы.*

Примеры текстовых констант: *'Ученик 8 «А» класса', 'Саша + Маша', '200$', '*****', '5 + 3<10'.*

Числовые константы – это:

1. Целые числа со знаком или без него. Например: 3458, –876, 37105.
2. Вещественные числа, в которых целая часть от дробной отделяется ***точкой***. Например: 3.5, –0.8713, 9801.003.

Целые и вещественные числа могут быть представлены в двух формах:

1. Обычная форма записи чисел. Например: –45, 7.004.
2. В нормализованной форме. Например:

а) число 47.561 можно записать как $4.7561 \cdot 10^1$, а в нормализованной форме запишется 4.7561E01;

б) число 0.00023 можно записать как $2.3 \cdot 10^{-4}$, а в нормализованной форме запишется 2.3E–04.

В общем виде нормализованную форму числа можно записать так: *mEn*, где *m* называется мантиссой числа, *n* называется порядком числа. Мантисса числа в своей целой части должна содержать одну цифру. Буква *E* заменяет число 10. Порядок числа – это степень числа 10.

Текстовым и числовым константам можно давать имена (идентификаторы). Например, константе *4.789* дадим имя *SUM*, а константе *'Учение'* – имя *B*. Тогда имена и значения этих констант нужно обязательно объявить в разделе описаний в подразделе, который начинается словом **CONST**, следующим образом:

Const *Sum=4.789;* *B='Учение';*

Типы констант определяются по их значениям и форме записи. Так, в нашем примере значение константы *Sum* – вещественное число, значит, и сама константа *Sum* – вещественного типа; значение константы *B* – набор символов, заключенный в апострофы, значит, константа *B* текстового типа. Если значение константы будет целое число, то и сама константа будет целого типа.

В языке **PASCAL** имеется одна служебная константа вещественного типа Pi. Это общепринятая математическая константа π. Также имеется одна константа целого типа maxint (значение этой константы равно 32767). Эти константы в разделе Const объявлять не нужно, так как они служебные.

Переменная – это величина, которая *может* менять свое значение при выполнении программы. Каждой переменной программист дает свое имя – идентификатор.

Все переменные, которые будут использоваться в программе должны быть объявлены в разделе описаний в подразделе, который начинается служебным словом **VAR**. После служебного слова идет список имен переменных. Имена переменных отделяются друг от друга запятой. После списка ставится двоеточие и записывается служебное слово, которое указывает тип указанных в списке переменных.

Простые типы, числовые интервалы, объем занимаемой памяти

В языке **PASCAL** имеется несколько типов для переменных, которые имеют целые и вещественные значения. Эти типы названы по-разному и отличаются друг от друга диапазоном значений и объемом занимаемой памяти. Все типы приведены в таблицах.

Таблица целочисленных типов переменных

Название типа	Служебное слово типа	Диапазон значений	Размер памяти
Целое со знаком	Integer	–32768 ... 32767	2 байта
Короткое целое со знаком	Shortint	–128 ... 127	1 байт
Длинное целое со знаком	Longint	–2147483648 ... 2147483647	4 байта
Короткое целое без знака	Byte	0 ... 255	1 байт
Целое без знака	Word	0 ... 65535	2 байта

Таблица вещественных типов переменных

Название типа	Служебное слово типа	Диапазон значений	Размер памяти
Вещественное число	Real	$2{,}9\cdot10^{-39}...1{,}7\cdot10^{38}$	6 байтов
Вещественное число с одинарной точностью	Single	$1{,}5\cdot10^{-45}...3{,}4\cdot10^{38}$	4 байта
Вещественное число с двойной точностью	Double	$5{,}0\cdot10^{-324}...1{,}7\cdot10^{308}$	8 байтов
Вещественное число с повышенной точностью	Extendet	$3{,}4\cdot10^{-4932}...1{,}1\cdot10^{4932}$	10 байтов
Вещественное сложное число	Comp	$-9{,}2\cdot10^{18}...9{,}2\cdot10^{18}$	8 байтов

В программе сначала объявляются константы, а потом переменные.

Структура программы

Программа на языке **PASCAL** состоит из строк. Максимальная длина строки не должна превышать 127 символов.

Размер программы имеет предел. Редактор текстов и компилятор позволяют обрабатывать программы и библиотечные модули объемом до 64 Кбайт. Если программа требует большего количества памяти, следует использовать библиотечные модули (.TPU-файлы).

В начале программы находится заголовок, состоящий в общем случае из зарезервированного слова **PROGRAM**, имени программы и параметров, с помощью которых программа взаимодействует с операционной системой. Заголовок программы несет чисто смысловую нагрузку и может отсутствовать, однако рекомендуется всегда его записывать для быстрого распознавания нужной программы среди листингов других программ. После заголовка следует программный блок, состоящий в общем случае из 7 разделов: списка имен подключаемых библиотечных модулей (он определяется зарезервированным словом **USES**), описания меток, описания констант, определения типов данных, описания переменных, описания процедур и функций, операторов.

Структура программы выглядит следующим образом:

```
PROGRAM <имя>;
USES <имя1, имя2,...>;
LABEL ...;
CONST ...;
TYPE ...;
VAR ...;
PROCEDURE <имя>;
<тело процедуры>
FUNCTION <имя>;
<тело функции>
BEGIN
<операторы>
END.
```

Любой раздел, кроме раздела операторов, может отсутствовать. Разделы описаний (кроме **USES**, который всегда расположен после заголовка программы) могут встречаться в программе любое количество раз и следовать в произвольном порядке. Главное, чтобы все описания объектов программы были сделаны до того, как они будут использованы.

Таблица стандартных функций

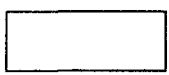
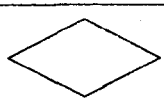
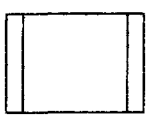
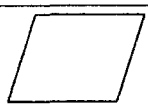
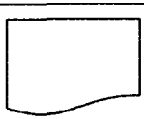
Обозначение функции	Тип аргумента x	Тип функции	Операция
Abs(x)	Целый, вещественный	Целая, вещественная	Вычисление абсолютного значения x
Sqr(x)	Целый, вещественный	Целая, вещественная	Вычисление x^2
Sqrt(x)	Целый, вещественный	Вещественная	Вычисление квадратного корня из x
Exp(x)	Целый, вещественный	Вещественная	Вычисление e^x
Frac(x)	Вещественный	Вещественная	Вычисление дробной части числа x
Int(x)	Вещественный	Вещественная	Вычисление целой части числа x
Trunc(x)	Вещественный	Целая	Нахождение целой части числа x по следующим правилам: 1) если $x >= 0$, то результат будет $<= x$; 2) если $x < 0$, то результат будет $> x$
Succ(x)	Целый	Целая	Выдает следующее за x значение
Pred(x)	Целый	Целая	Выдает предшествующее x значение
Ln(x)	Целый, вещественный	Вещественная	Вычисление Ln(x)
Round(x)	Вещественный	Целая	Округлить до ближайшего целого числа
Sin(x)	Целый	Вещественная	Вычисление Sin(x)
Cos(x)	Вещественный	Вещественная	Вычисление Cos(x)
Arctan(x)	Целый	Вещественная	Вычисление Arctg(x)
Odd(x)	Целый	Логическая	Выдает значение «Истина», если число x нечетное; значение «Ложь», если x – четное

Графический способ представления алгоритмов

К этому способу относят блок-схемы, граф-схемы, структурограммы. Рассмотрим представление алгоритмов в виде блок-схем. При таком

способе представления алгоритма каждый шаг алгоритма представляется геометрической фигурой, внутри которой записана команда. Такие геометрические фигуры называются блоками. Для указания порядка исполнения блоков используются стрелки. Таким образом, под *блок-схемой будем понимать графическое представление последовательности шагов алгоритма, которое наглядно показывает очередность и взаимосвязь операций, осуществляемых в алгоритме на каждом его шаге.* Иначе говоря, блок-схема служит для графического представления структуры алгоритма.

Основные типы блоков

	*Начало и конец алгоритма* (для функций «Вход», «Выход»)
	*Блок обработки.* Внутри блока записываются формулы, обозначения и функции
	*Блок условия.* Внутри блока записываются условия выбора направления действия алгоритма
	*Блок предопределенного процесса* (функция/подпрограмма)
	*Блок ввода информации*
	*Блок цикла с известным количеством повторений*
	*Блок ввода информации на печатающее устройство*
	*Соединительный блок*

Пример разработки линейной программы

1. Составить программу по вычислению.

$$t = \left(\sin^2 x + \left(\cos x - \frac{\sin x + x^2}{x} \right)^3 + 10,2x \right)^4 .$$

2. Решение.

В языке Паскаль нет операции возведения в степень, поэтому возведение в степень осуществляется следующим образом:

$$a^x = \exp(x * \mathrm{Ln}(a)) .$$

Представим вычисление по приведенной выше формуле в виде

$$a = \cos x - \frac{\sin x + x^2}{x};$$

$$b = \sin^2 x + a^3 + 10,2 * x;$$

$$t = b^4 .$$

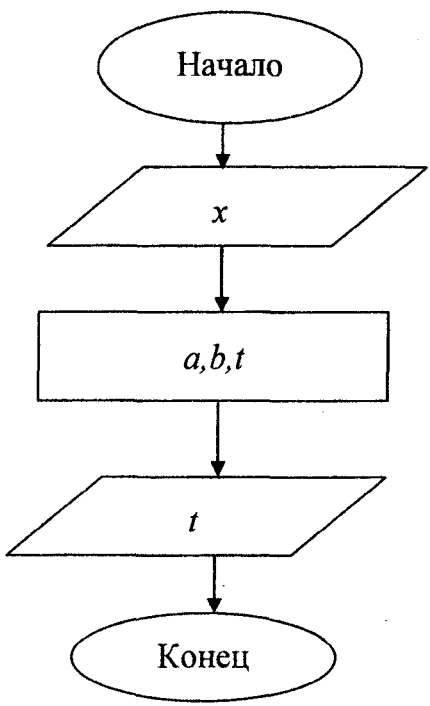
Это позволит значительно упростить программирование заданного выражения.

Текст исходной программы:

```
Program Calc;
Uses Crt;
Var
a,b,t,x:Real;
Begin
```

```
Clrscr;
writeln ('x= ');
readln (x);
a:= cos x – (sin x + sqr (x)) / x;
b:= sin x * sin x + exp(3 * Ln(a)) + 10,2 * x;
t:=exp(4 *Ln(b));
writeln;
write('t=',t:10:4);
repeat until keypressed;
End.
```

Схему алгоритма решения задачи представим в виде



Варианты заданий

Номер варианта	Выражение	Исходные данные
1	2	3
1	$a = \ln(y^{-\sqrt{\lvert x\rvert}}) \cdot \left(\sin(x) + e^{(x+y)}\right)$	*x, y*
2	$b = \sqrt{c(\sqrt{y} + x^2)} \cdot (\cos(x) - \lvert c - y\rvert)$	*c, x, y*
3	$c = \operatorname{arctg}(x) - \frac{3}{5} e^{xy} + 0{,}5 \frac{\lvert x + y\rvert}{(x+y)^b}$	*b, x, y*
4	$d = \frac{e^{\lvert x-y\rvert} \cdot \operatorname{tg}(z)}{\operatorname{arctg}(y) + \sqrt{x}} + \ln(x)$	*x, y, z*
5	$e = \frac{(\cos(x) - \sin(y))^3}{\sqrt{\operatorname{tg}(z)}} + \ln^2(x \cdot y \cdot z)$	*x, y, z*
6	$f = y^x + \sqrt{\lvert x\rvert + e^y} - \frac{z^3 \cdot \sin^2(y)}{y + z^2/(y - x)}$	*x, y, z*
7	$g = \frac{1 + \cos(x+y)}{\lvert e^x - 2y/(1 + x^2 \cdot y^2)\rvert} \cdot x^3 + \arcsin(y)$	*x, y*
8	$h = 2 + \frac{x^2}{\sqrt{2}} + \frac{\lvert y^3\rvert}{\sqrt{2}} + \frac{z^4 \cdot (\ln(x) + 1) \cdot \sqrt{2}}{\sqrt{3}}$	*x, y, z*
9	$j = \left((1 + y) \cdot \sqrt{\sin^2(z)} - \frac{\lvert y - x\rvert}{5} \right)^3$	*x, y, z*
10	$k = \ln\lvert y - \sqrt{\lvert x\rvert}\rvert \cdot (x - \frac{y}{z + x^2/4})$	*x, y, z*
11	$l = 0{,}5x^2 + 3 \cdot \cos(x + y) + e^{-0{,}1y \cdot z} - \sqrt{\lvert x \cdot y\rvert}$	*x, y, z*
12	$m = \sqrt{e^x + \operatorname{tg}(x) + 1} \cdot (\lg(y) + \cos(x \cdot y) + \sqrt[3]{x})$	*x, y*

1	2	3
13	$p = \dfrac{\lg(x) - e^{x+y}}{\sqrt{2 + y^2 + \lvert x^3 - \ln(y) \rvert}}$	*x, y*
14	$q = \sqrt{12x^4 - 3x^2 + 4x^2 - 5x + 6} - \lg^2(z)$	*x, z*
15	$\alpha = \dfrac{ax^2 + \sin^2 z}{\sqrt{1 + e^a}}$	*a, x, z*
16	$t = \dfrac{\beta^2 + \sqrt{\lvert q \rvert}}{\cos^2 x + \beta \ln x}$	β, *z, x*
17	$q = \dfrac{\sin^2(z + a)^3}{t\sqrt[3]{e^{2+3t}}}$	*z, a, t*

Контрольные вопросы

1. Как описываются константы и переменные?
2. Назвать стандартные типы данных.
3. Описать процедуру стандартного ввода-вывода.
4. Назвать составные части программы на языке **PASCAL**.
5. Изобразить схематически блок-схему линейного алгоритма.

Лабораторная работа № 6

ВЫЧИСЛЕНИЕ ПО УСЛОВИЮ

Цель работы: изучение оператора **IF** при решении задач на языке **PASCAL** и представление алгоритма решения задачи в виде блок-схемы с использованием структуры ветвления.

Теоретические сведения

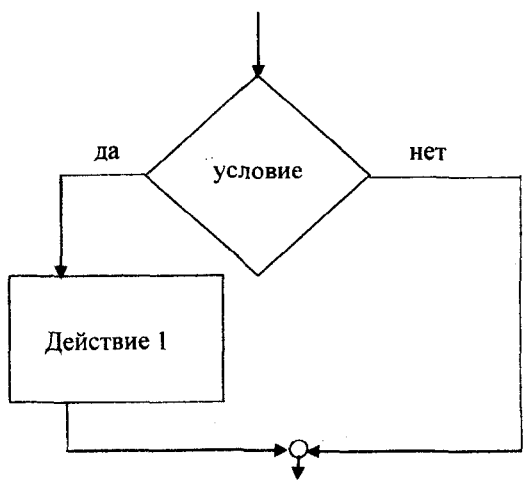
В языке **PASCAL** различают два вида условных операторов: короткий и полный.

Короткий условный оператор

Общий вид записи:
if <условие> then <оператор>.

Графическая интерпретация оператора

В блок-схемах короткому условному оператору соответствует структура **ЕСЛИ-ТО**.



Отметим, что данная структура имеет один вход и один выход. Словесная формулировка данной структуры: «Если условие истинно, то выполнять Действие 1».

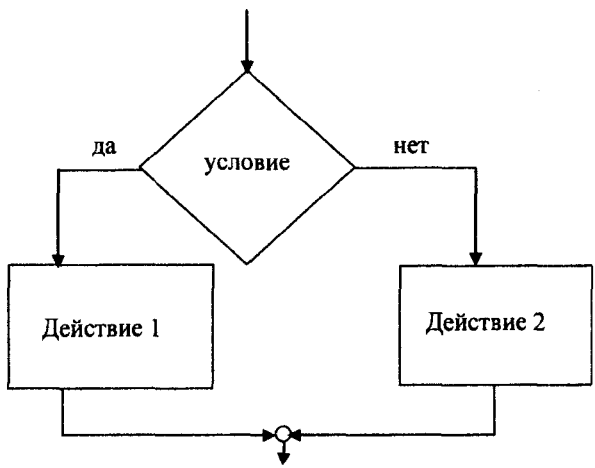
Полный условный оператор

Общий вид записи:

if <условие> then <оператор 1>
else <оператор 2>;

Графическая интерпретация оператора

В блок-схемах полному условному оператору соответствует структура **ЕСЛИ-ТО-ИНАЧЕ**.



Отметим, что и данная структура имеет один вход и один выход. Словесная формулировка данной структуры: «Если значение выражения истинно, выполняется <действие 1>, если ложно – <действие 2>».

Замечание. Оператор 1 и оператор 2 входят в конструкцию полного условного оператора как единственные (**простые**). Если возникает необходимость выполнить в ветвях несколько операторов, то их заключают в операторные скобки begin... end. Такие операторы называются **составными**. Запись оператора **IF** с **составным оператором** имеет вид

```
if <логическое выражение> then
    begin
        <оператор 1>;
.....
<оператор n>;
else
    begin
        <оператор 1>;
.....
        <оператор m>;
end;
```

Условие – это выражение булевского типа. Оно может быть простым и сложным. **Сложные условия** образуются с помощью логических операций **and**, **or**, **not**.

Вложенная структура условных операторов

Структура называется **вложенной**, если после служебных слов **THEN** и **ELSE** используются вновь условные операторы. Число вложений может быть произвольным. При этом справедливо: служебное слово **ELSE** всегда относится к ближайшему выше слову **THEN**.

```
If <условие> then
If <условие> then <оператор>
                  else <оператор>;
```

Ниже приведены типичные формы использования оператора **IF**.

1. Короткая форма записи оператора IF:

```
A:=2; B:=8; C:=0;
If A<B then C:=A+B;
writeln('C=', C:2);
```

2. Полная форма записи оператора IF:

а) с простым оператором:

```
If A>B then D:=A+B else D:=A-B;
```

б) с составным оператором:

```
If A<B then
        begin
          writeln('A меньше B');
          D:=A*B
        end
          else begin
               writeln('A больше или равно B');
          D:=A/B
        end;
```

3. С использованием сложного условия:

```
If (A=B) and (C=D) then
begin
    writeln('Норма'); F:=0
end
  else
    begin
    writeln('Превышение нормы!'); F:=100
end;
```

4. С использованием вложенности оператора:

```
If A<20 then
If A>=15 then writeln('A в диапазоне 15-20')
          else writeln('A в диапазоне 10-14');
```

Варианты заданий

Номер варианта	Выражение	Исходные данные
1	2	3
1	$a=\begin{cases}(x+y)^2-\sqrt{x\cdot y}, & x\cdot y>0\\(x+y)^2+\sqrt{\lvert x\cdot y\rvert}, & x\cdot y<0\\(x+y)^2+1, & x\cdot y=0\end{cases}$	x, y

1	2	3
2	$b=\begin{cases}\ln(x/y)+(x^2+y)^3, & x/y>0\\ \ln\lvert x/y\rvert+(x^2+y)^3, & x/y<0\\ (x^2+y), & x=0\\ 0, & y=0\end{cases}$	*x, y*
3	$c=\begin{cases}x^2+y^2+\sin(x), & x-y=0\\ (x-y)^2+\cos(x), & x-y>0\\ (y-x)^2+\operatorname{tg}(x), & x-y<0\end{cases}$	*x, y*
4	$d=\begin{cases}(x-y)^3+\operatorname{arctg}(x), & x>y\\ (y-x)^3+\operatorname{arctg}(x), & y>x\\ (y+x)^3+0{,}5, & y=x\end{cases}$	*x, y*
5	$e=\begin{cases}i\cdot\sqrt{a}, & i-\text{нечетное}, a>0\\ \frac{i}{2}\sqrt{\lvert a\rvert}, & i-\text{четное}, a<0\\ \sqrt{\lvert i\cdot a\rvert}, & \text{иначе}\end{cases}$	*i, a*
6	$g=\begin{cases}e^{\lvert a\rvert-\lvert b\rvert}, & 0{,}5<a\cdot b<10\\ \sqrt{\lvert a+b\rvert}, & 0{,}1<a\cdot b<0{,}5\\ 2x^2, & \text{иначе}\end{cases}$	*a, b, x*
7	$h=\begin{cases}\operatorname{arctg}(x+\lvert y\rvert), & x<y\\ \operatorname{arctg}(\lvert x\rvert+y), & x>y\\ (x+y)^2, & x=y\end{cases}$	*x, y*
8	$j=\begin{cases}\sin(5k+3\cdot m\cdot\lvert k\rvert), & -1<k<m\\ \cos(5k+3\cdot m\cdot\lvert k\rvert, & k>m\\ k^3, & k=m\end{cases}$	*k, m*
9	$l=\begin{cases}3k^3+3p, & k>\lvert p\rvert\\ \lvert k-p\rvert, & 3<k<\lvert p\rvert\\ (k-p)^2, & k=\lvert p\rvert\end{cases}$	*k, p*

1	2	3
10	$k=\begin{cases}\ln(\lvert f\rvert+\lvert q\rvert), & \lvert f\cdot q\rvert>10\\ e^{f+q}, & \lvert f\cdot q\rvert<10\\ f+q, & \lvert f\cdot q\rvert=10\end{cases}$	*f, q*
11	$y=\begin{cases}ax^3+b\ln\lvert 2x\rvert, & \sqrt{\lvert a-b\rvert}<x\\ \sqrt{\lvert a+\sin 2x\rvert}-b^{3x}, & \sqrt{\lvert a-b\rvert}=x\\ \dfrac{\operatorname{arctg}5x}{b\cos x+\lg ax}, & \sqrt{\lvert a-b\rvert}>x\end{cases}$	*a, b, x*
12	$y=\begin{cases}\ln x^2-e^{ax+b}+\lg\lvert a-b\rvert, & 2a+b<0\\ \operatorname{tg}(ax-b^2)-b\lvert x^{-3}\rvert, & 2a+b=0\\ \operatorname{arctg}(2x-0{,}5)+\sqrt{\lvert a+bx\rvert}, & 2a+b>0\end{cases}$	*a, b, x*
13	$y=\begin{cases}\dfrac{ax-e^x+b^3}{\ln(2x^2+4)}+\cos(4x-0{,}2), & \lvert a^2-b^2\rvert<10x\\ b\sin(x^3-a)-e^{-1,4x}, & \lvert a^2-b^2\rvert=10x\\ tg4x+\dfrac{\lvert x^{-5}\rvert}{\sin 0{,}5x}, & \lvert a^2-b^2\rvert>10x\end{cases}$	*a, b, x*
14	$y=\begin{cases}a\sqrt{\lvert\sin x+\cos^2 x\rvert}+e^{a+bx}, & \sqrt{\lvert bx-65\rvert}<a\\ 1-\ln\sqrt{\lvert ax^3-b\rvert}-\lvert x^{-10}\rvert, & \sqrt{\lvert bx-65\rvert}=a\\ \dfrac{(x^3-b)\cdot\cos(3x-0{,}5)}{\lg x^3-a\sin(a-b)}, & \sqrt{\lvert bx-65\rvert}>a\end{cases}$	*a, b, x*
15	$y=\begin{cases}\ln\left\lvert\dfrac{x^{-9}}{ax-b}\right\rvert-e^{2x^2}, & \sin x<\sqrt{a^2+b^2}\\ 4\operatorname{arctg}6x+\dfrac{ax+7}{\sqrt{\lvert b^2-x\rvert}}, & \sin x=\sqrt{a^2+b^2}\\ x^3+2{,}3ax+\sqrt{b^2\lvert\operatorname{tg}x\rvert}, & \sin x>\sqrt{a^2+b^2}\end{cases}$	*a, b, x*

1	2	3
16	$\alpha = \begin{cases} \sqrt{\sin x^3 + \ln^2 z}, & \text{при } z < \sqrt[3]{ax} \\ e^{\beta \cdot z} + \lvert x^3 \rvert, & \text{при } z = \sqrt[3]{ax} \\ \cos\sqrt{z} + \lg \beta^2 x, & \text{при } z > \sqrt[3]{ax} \end{cases}$	x, z, a, β
17	$\beta = \begin{cases} 3q + \sqrt{\lvert x^3 + \sin z \rvert}, & \text{при } \sin z < q \\ \sqrt[5]{\alpha^3 + q^3 + e^{\alpha \cdot z}}, & \text{при } \sin z = q \\ \lvert \alpha^5 \rvert \cdot \ln\sqrt{\cos z}, & \text{при } \sin z > q \end{cases}$	q, x, z, α
18	$\gamma = \begin{cases} \alpha^3 + \sqrt[5]{\cos y^2}, & \text{при } y < \ln\beta \\ \lg(x+w)^2 + \lvert y^3 \rvert, & \text{при } y = \ln\beta \\ \sqrt{\operatorname{tg} y^5} + e^{xw}, & \text{при } y > \ln\beta \end{cases}$	α, y, β, x, w

Контрольные вопросы

1. Формат записи оператора **IF**.
2. Формат записи с вложенным оператором **IF**.
3. Как работает оператор **IF**?
4. В чем отличие короткой и полной формы записи оператора **IF**?
5. Изобразить схематически блок-схему разветвляющего алгоритма.

Лабораторная работа № 7

ОПЕРАТОР ВЫБОРА CASE

Цель работы: изучение оператора **CASE** при решении задач на языке **PASCAL** и представление алгоритма решения задачи в виде блок-схемы с использованием структуры ветвления.

Теоретические сведения

Оператор выбора **CASE** позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого селектором, и списка параметров, каждому из которых предшествует список констант выбора (список может состоять и из одной константы).

Формат:
CASE <выражение-селектор> OF
<список1>: <оператор1; >
<список2>: <оператор2; >
. . .
<списокN>: <операторN>
ELSE <оператор>
END;

Тип констант в любом случае должен быть совместим с типом селектора. Для селектора запрещены типы real и string.

Оператор **CASE** работает следующим образом. Сначала вычисляется значение выражения-селектора, затем обеспечивается реализация того оператора, константа выбора которого равна текущему значению селектора. Если ни одна из констант не равна текущему значению селектора, выполняется оператор, стоящий за словом **ELSE**. Если слово **ELSE** отсутствует, активизируется оператор, находящийся за словом **END**, т. е. первый оператор за границей **CASE**.

Селектор должен относиться к одному из целочисленных (находящихся в диапазоне –32768…32767) булевскому, литерному или пользовательскому типов.

Список констант выбора состоит из произвольного количества значений или диапазонов, отделенных друг от друга запятыми. Границы диапазона записываются двумя константами через разграничитель «..». Тип констант в любом случае должен совпадать с типом селектора.

Ниже приведены типичные формы записи оператора **CASE**.

Селектор интервального типа:

```
CASE I OF
1..10: Writeln ('число ', I:4, ' в диапазоне 1 - 10');
11..20: Writeln ('число ', I:4, ' в диапазоне 11 - 20');
21..30: Writeln ('число ', I:4, ' в диапазоне 21 - 30')
ELSE Writeln ('число ', I:4, ' вне пределов контроля')
END;
```

Селектор целочисленного типа:

```
CASE I OF
1: Z:=I + 10;
2: Z:=I + 100;
3: Z:=I + 1000
END;
```

Селектор перечисляемого пользовательского типа:

```
  VAR
    Season: (Winter, Spring, Summer, Autumn);
  BEGIN
  ...
  CASE Season OF
    Winter: Writeln('Winter');
    Spring: Writeln('Spring');
    Summer: Writeln('Summer');
    Autumn: Writeln('Autumn')
  END;
END;
```

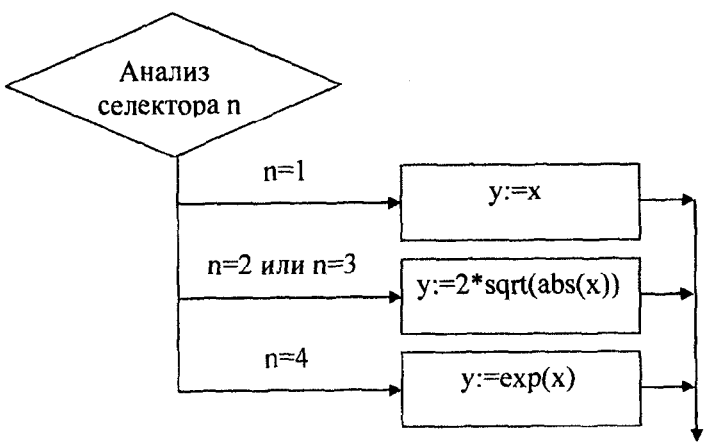
Пример.
Написать оператор выбора *n* для вычисления *y*

$$y = \begin{cases} x, & \text{если} \quad n = 1; \\ 2\sqrt{|x|}, & \text{если} \quad n = 2 \quad \text{или} \quad n = 3; \\ e^x, & \text{если} \quad n = 4. \end{cases}$$

```
case n of
    1: y:=x;
  2,3: y:=2*sqrt(abs(x));
    4: y:=exp(x);
end;
```

Графическая интерпретация оператора CASE

В блок-схемах оператору CASE соответствует структура ВЫБОР.



Замечание. Если в строке выбора необходимо записать несколько операторов, то их заключают в операторные скобки **BEGIN...END**.

Варианты заданий

Номер варианта	Выражение	Исходные данные
1	2	3
1	$a=\begin{cases}(x+y)^2-\sqrt{x\cdot y}, & k=1\\ (x+y)^2+\sqrt{\lvert x\cdot y\rvert}, & k=12\\ (x+y)^2+1, & k=100\end{cases}$	x, y, k
2	$b=\begin{cases}\ln(x/y)+(x^2+y)^3, & \alpha=a\div d\\ \ln\lvert x/y\rvert+(x^2+y)^3, & \alpha=k\div p\\ (x^2+y), & \alpha=r\div y\\ 0, & \text{иначе}\end{cases}$	x, y, α
3	$c=\begin{cases}x^2+y^2+\sin(x), & \beta=10,12,23\\ (x-y)^2+\cos(x), & \beta=14,18,24\\ (y-x)^2+\operatorname{tg}(x), & \beta=100,120,235\end{cases}$	x, y, β
4	$d=\begin{cases}(x-y)^3+\operatorname{arctg}(x), & z=1\div 10\\ (y-x)^3+\operatorname{arctg}(x), & z=12\div 34\\ (y+x)^3+0{,}5, & \text{иначе}\end{cases}$	x, y, z
5	$e=\begin{cases}i\cdot\sqrt{a}, & \lambda=1,2,3\\ \frac{i}{2}\sqrt{\lvert a\rvert}, & \lambda=5,6,7\\ \sqrt{\lvert i\cdot a\rvert}, & \text{иначе}\end{cases}$	i, a, λ
6	$g=\begin{cases}e^{\lvert a\rvert-\lvert b\rvert}, & s=1\\ \sqrt{\lvert a+b\rvert}, & s=12\\ 2x^2, & \text{иначе}\end{cases}$	a, b, x, s
7	$h=\begin{cases}\operatorname{arctg}(x+\lvert y\rvert), & \eta=1\\ \operatorname{arctg}(\lvert x\rvert+y), & \eta=12,\ 23,56\\ (x+y)^2, & \text{иначе}\end{cases}$	x, y, η

1	2	3
8	$j=\begin{cases}\sin(5k+3\cdot m\cdot\lvert k\rvert), & \alpha=a\div d\\ \cos(5k+3\cdot m\cdot\lvert k\rvert), & \alpha=k\div t\\ k^3, & \text{иначе}\end{cases}$	*k, m,* α
9	$l=\begin{cases}3k^3+3p, & w=12\\ \lvert k-p\rvert, & w=34\\ (k-p)^2, & w=89\end{cases}$	*k, p, w*
10	$k=\begin{cases}\ln(\lvert f\rvert+\lvert q\rvert), & h=1\div 23\\ e^{f+q}, & h=34\div 89\\ f+q, & h=90\div 123\end{cases}$	*f, q, h*
11	$y=\begin{cases}ax^3+b\ln\lvert 2x\rvert, & m=12\div 45\\ \sqrt{\lvert a+\sin 2x\rvert}-b^{3x}, & m=46\div 67\\ \dfrac{\operatorname{arctg}5x}{b\cos x+\lg ax}, & \text{иначе}\end{cases}$	*a, b, x, m*
12	$y=\begin{cases}\ln x^2-e^{ax+b}+\lg\lvert a-b\rvert, & \tau=123\div 234\\ \operatorname{tg}(ax-b^2)-b\lvert x^{-3}\rvert, & \tau=236\div 345\\ \operatorname{arctg}(2x-0{,}5)+\sqrt{\lvert a+bx\rvert}, & \tau=1000\end{cases}$	*a, b, x,* τ
13	$y=\begin{cases}\dfrac{ax-e^x+b^3}{\ln(2x^2+4)}+\cos(4x-0{,}2), & z=1\\ b\sin(x^3-a)-e^{-1,4x}, & z=56\\ \operatorname{tg}4x+\dfrac{\lvert x^{-5}\rvert}{\sin 0{,}5x}, & z=123\end{cases}$	*a, b, x, z*
14	$y_1=\begin{cases}a\sqrt{\lvert\sin x+\cos^2 x\rvert}+e^{a+bx}, & y=12,45,78\\ 1-\ln\sqrt{\lvert ax^3-b\rvert-\lvert x^{-10}\rvert}, & y=456\\ \dfrac{(x^3-b)\cdot\cos(3x-0{,}5)}{\lg x^3-a\sin(a-b)}, & \text{иначе}\end{cases}$	*a, b, x, y*

1	2	3
15	$y=\begin{cases}\ln\left\|\dfrac{x^{-9}}{ax-b}\right\|-e^{2x^2}, & s=34\div 56\\ \operatorname{arctg}6x+\dfrac{ax+7}{\sqrt{\left\|b^2-x\right\|}}, & s=57\div 78\\ x^3+2{,}3ax+\sqrt{b^2\left\|\operatorname{tg}x\right\|}, & \text{иначе}\end{cases}$	a, b, x, s
16	$a=\begin{cases}\sqrt{\sin x^3}+\ln^2 z, & \alpha=12\div 34\\ e^{\beta\cdot z}+\left\|x^3\right\|, & \alpha=35\div 56\\ \cos\sqrt{z}+\lg\beta^2 x, & \text{иначе}\end{cases}$	x, z, α, β
17	$b=\begin{cases}3q+\sqrt{\left\|x^3+\sin z\right\|}, & \beta=a\div c\\ \sqrt[5]{\alpha^3+q^3+e^{\alpha\cdot z}}, & \beta=d\div m\\ \left\|\alpha^5\right\|\cdot\ln\sqrt{\cos z}, & \text{иначе}\end{cases}$	q, x, z, α, β
18	$d=\begin{cases}\alpha^3+\sqrt[5]{\cos y^2}, & \gamma=1\\ \lg(x+w)^2+\left\|y^3\right\|, & \gamma=12{,}45{,}89\\ \sqrt{\operatorname{tg}y^5}+e^{xw}, & \text{иначе}\end{cases}$	α, y, x, w, γ

Контрольные вопросы

1. Формат записи оператора **CASE**.
2. Как работает оператор **CASE**?
3. В чем отличие оператора **CASE** от **IF**?
4. Типы селектора.
5. Изобразить схематически блок-схему разветвляющего алгоритма.

Лабораторная работа № 8

ТАБУЛИРОВАНИЕ ФУНКЦИИ

Выполнить лабораторную работу 3 способами:
1) с использованием оператора **FOR**;
2) с использованием оператора **WHILE**;
3) с использованием оператора **REPEAT UNTIL**.

Цель работы: изучение всех видов операторов цикла при решении задач на языке **PASCAL** и представление алгоритма решения задачи в виде блок-схемы.

Теоретические сведения

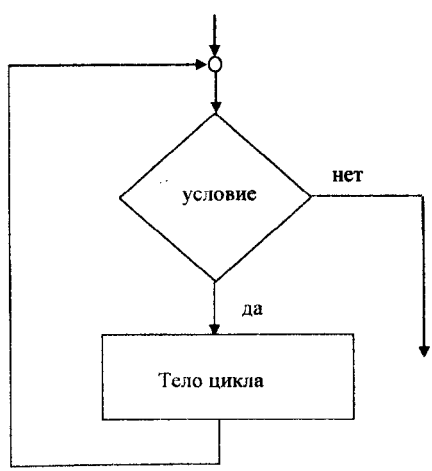
Зачастую в программах необходимо отдельный оператор или группу операторов повторять многократно. Чтобы их не записывать в программе несколько раз, существуют операторы повторений (циклов).

Любой цикл всегда состоит из операторов, которые повторяются, и условия, которое позволяет на некотором шаге выйти из цикла. Операторы, которые повторяются, образуют ***тело цикла***. Условие, по которому образуется выход из цикла, является ***условием цикла***.

В языке Pascal имеются три формы операторов цикла: **WHILE**, **FOR**, **REPEAT.**

1. Оператор повтора с предусловием WHILE

Схематически оператор повтора с предусловием изображается так:



Формат записи оператора повтора с предусловием **WHILE**:

```
WHILE условие
    Тело цикла;
```

WHILE – служебное слово – Пока, ***условие*** – логическое выражение, которое может принимать значение True или False, ***тело цикла*** – любые операторы языка **PASCAL**. Если тело цикла состоит из двух и более операторов, то оно заключается в операторные скобки **BEGIN – END**.

Оператор цикла работает следующим образом: пока условие принимает значение ***True***, выполняется тело цикла. Выход из цикла происходит, когда условие примет значение ***False***.

Оператор цикла **WHILE** еще называют оператором цикла ***пока*** (по способу работы цикла) или оператором цикла ***с предусловием***, так как сначала в операторе записано условие и проверяется его истинность, а затем уже выполняется или нет тело цикла.

Примеры использования оператора WHILE

Пример 1. Числа Фибоначчи строятся по следующим формулам: $a_1 = 1$, $a_2 = 1$, $a_n = a_{n-1} + a_{n-2}$. Получить *n-е* число Фибоначчи.

Программа на языке **PASCAL**:

```
Program Fibonachi;
Var n,a,b,c,k:Longint;
Begin
   Writeln('Ввести номер числа');
   Readln(n);
   a:=1;
   b:=1;
   k:=2;
   While k<n do
      Begin c:=a+b;
           a:=b;
           b:=c;
           k:=k+1;
      End;
If n<=2
      Then Writeln(n:3,'-е число Фибоначчи=1')
      Else Writeln(n:3,'-е число Фибоначчи=',c:7);
End.
```

2. Оператор повтора с известным числом повторений FOR

Данная структура используется только в том случае, если известно заранее количество повторений тела цикла. Количество повторений может задаваться как промежуток между начальным и конечным значением переменной, которая может изменять свое значение на 1. Эта переменная называется счетчиком или переменной цикла.

Формат записи оператора повтора с известным числом повторений **FOR**:

FOR *счетчик:=начальное значение* **TO** *конечное значение* **DO** *тело цикла;*

Приведенный оператор работает следующим образом. Сначала счетчику присваивается начальное значение. Если это значение меньше или равно конечному значению, то выполняется тело цикла. Потом значение счетчика увеличивается на 1 и опять сравнивается с конечным значением и т. д. Когда значение счетчика станет больше конечного значения, происходит выход из цикла. Учитывая сказанное, можно сказать, что если с самого начала значение счетчика больше конечного значения, то тело цикла не выполнится ни разу.

В языке **PASCAL** счетчик может уменьшаться на 1. Тогда оператор **FOR** записывается следующим образом:

FOR *счетчик:=начальное значение* **DOWNTO** *конечное значение* **DO** *тело цикла;*

Если тело цикла состоит не из одного оператора, то оно заключается в операторные скобки **BEGIN – END**.

Примеры использования оператора FOR

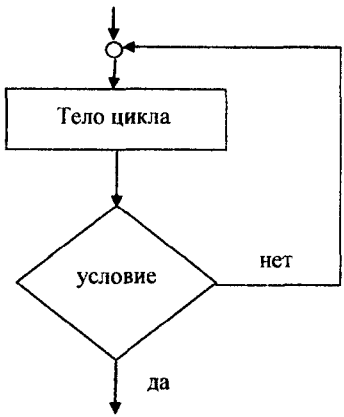
Пример 2. Вычислить факториал целого числа n. N!=1*2*3*…*n.

Программа на языке **PASCAL**:

```
Program Fact;
Var i,n:Integer; p:Longint;
Begin
   Writeln('Ввести значение n');
   Readln(n);
   p:=1;
   For i:=1 to n do p:=p*i;
   Writeln(n:4,'!=',p:10);
End.
```

3. Оператор повтора с постусловием REPEAT

Схематически такую структуру можно представить в следующем виде:



Данная структура на языке **PASCAL** имеет следующий формат записи:

REPEAT *тело цикла* **UNTIL** *условие*

Замечание. Так как тело цикла находится между служебными словами **REPEAT** и **UNTIL**, то заключать тело цикла в операторные скобки не нужно.

Примеры использования оператора REPEAT

Пример 3. Вывести на экран таблицу значений функции $y = e^x + \sin|x|$ на отрезке $[a, b]$ с шагом h. Значения a, b, h вводятся с клавиатуры.

```
Program Tabl;
Var a,b,h,x,y:Real;
Begin
   Writeln('Ввести начало отрезка');
Readln(a);
   Writeln('Ввести конец отрезка');
```

```
Readln(b);
   Writeln('Ввести шаг изменения x');
Readln(h);
x:=a;
Repeat
     y:=exp(x)+sin(abs(x));
     Writeln('x=',x:6:2,' y=',y:8:2);
     x:=x+h;
   Until x>b;
End.
```

Замечание. Все программы, которые можно создать с помощью цикла **REPEAT**, можно создать и с помощью цикла **WHILE**. А вот наоборот не всегда возможно, так как в программах с циклом **REPEAT** тело цикла должно выполниться хотя бы один раз, а в циклах **WHILE** тело цикла может не выполниться ни разу.

Варианты заданий

Номер варианта	Выражение	Исходные данные
1	2	3
1	$a = \ln(y^{-\sqrt{\lvert x\rvert}}) \cdot \left(\sin(x) + e^{(x+y)}\right)$	$x_н < x < x_к$, y = const
2	$b = \sqrt{c(\sqrt{y} + x^2)} \cdot (\cos(x) - \lvert c - y\rvert)$	$x_н < x < x_к$, y, c = const
3	$c = \operatorname{arctg}(x) - \frac{3}{5}e^{xy} + 0{,}5\frac{\lvert x+y\rvert}{(x+y)^b}$	$x_н < x < x_к$, y, b = const
4	$d = \frac{e^{\lvert x-y\rvert} \cdot tg(z)}{\operatorname{arctg}(y) + \sqrt{x}} + \ln(x)$	$x_н < x < x_к$, y, z = const
5	$e = \frac{(\cos(x) - \sin(y))^3}{\sqrt{tg(z)}} + \ln^2(x \cdot y \cdot z)$	$x_н < x < x_к$, y, z = const
6	$f = y^x + \sqrt{\lvert x\rvert + e^y} - \frac{z^3 \cdot \sin^2(y)}{y + z^2/(y-x)}$	$x_н < x < x_к$, y, z = const
7	$g = \frac{1 + \cos(x+y)}{\lvert e^x - 2y/(1 + x^2 \cdot y^2)\rvert} \cdot x^3 + \arcsin(y)$	$x_н < x < x_к$, y = const

1	2	3
8	$h = 2 + \frac{x^2}{\sqrt{2}} + \frac{\lvert y^3 \rvert}{\sqrt{2}} + \frac{z^4 \cdot (\ln(x)+1) \cdot \sqrt{2}}{\sqrt{3}}$	$x_н < x < x_к$, y, z = const
9	$j = \left((1+y) \cdot \sqrt{\sin^2(z)} - \frac{\lvert y - x \rvert}{5} \right)^3$	$x_н < x < x_к$, y, z = const
10	$k = \ln\left\lvert y - \sqrt{\lvert x \rvert} \right\rvert \cdot (x - \frac{y}{z + x^2/4})$	$x_н < x < x_к$, y, z = const
11	$l = 0{,}5x^2 + 3 \cdot \cos(x+y) + e^{-0{,}1y \cdot z} - \sqrt{\lvert x \cdot y \rvert}$	$x_н < x < x_к$, y, z = const
12	$m = \sqrt{e^x + tg(x) + 1} \cdot (\lg(y) + \cos(x \cdot y) + \sqrt[3]{x})$	$x_н < x < x_к$, y = const
13	$p = \frac{\lg(x) - e^{x+y}}{\sqrt{2 + y^2 + \lvert x^3 - \ln(y) \rvert}}$	$x_н < x < x_к$, y = const
14	$p = \frac{\lg(x) - e^{x+y}}{\sqrt{2 + y^2 + \lvert x^3 - \ln(y) \rvert}}$	$x_н < x < x_к$, y = const
15	$q = \sqrt{12x^4 - 3x^2 + 4x^2 - 5x + 6} - \lg^2(z)$	$x_н < x < x_к$, z = const
16	$\alpha = \frac{ax^2 + \sin^2 z}{\sqrt{1 + e^a}}$	$x_н < x < x_к$, a, z = const
17	$t = \frac{\beta^2 + \sqrt{\lvert q \rvert}}{\cos^2 x + \beta \ln x}$	$x_н < x < x_к$, β, q = const
18	$q = \frac{\sin^2(z+a)^3}{t\sqrt[3]{e^{2+3t}}}$	$t_н < t < t_к$, a, z = const

Контрольные вопросы

1. Формат записи оператора **FOR**.
2. Формат записи оператора **WHILE**.
3. Формат записи оператора **REPEAT UNTIL**.
4. В чем отличие операторов **FOR** и **WHILE**?
5. Составить блок-схему циклического алгоритма для всех операторов цикла.

Лабораторная работа № 9

ВЫЧИСЛЕНИЕ КОНЕЧНЫХ СУММ

Цель работы: изучение вложенных структур операторов повтора (циклов) при вычислении конечных сумм на языке **PASCAL** и представление алгоритма решения задачи в виде блок-схемы.

Теоретические сведения

Оператор цикла часто применяется для суммирования значений некоторой последовательности чисел или значений функции при известном числе операций суммирования. Напомним некоторые определения, связанные с расчетом суммы последовательности.

Сумма членов последовательности величин a_1, a_2, a_3, …, a_N называется конечной суммой $S_N = a_1 + a_2 + a_3 + ... + a_N$. Для некоторых последовательностей известны формулы расчета конечных сумм, например:

при $a_N = a_{N-1} + d$ $S_N = (a_1 + a_N)*N/2$ – арифметическая прогрессия;

при $a_N = a_{N-1} * q$ $S_N = (a_1 - a_N*q)/(1 - q)$ – геометрическая прогрессия,

где d и q – постоянные числа.

Здесь N-й член последовательности выражается через $(N–1)$-й член. Такие зависимости называются реккурентными.

Конечная сумма последовательности может быть неизвестна, тогда для ее расчета применяется алгоритм суммирования членов последовательности в цикле от 1 до N. Приведем пример расчета конечной суммы последовательности:

$$1^2 + 3^2 + 5^2 + ... + (2*N - 1)^2;\ S_N = N*(4*N^2 - 1)/3;$$

```
PROGRAM SUM_K;                          {расчет конечной суммы}
var a, S, Sn, i, N: word;
Begin
write('Введите число членов суммы N='); readln(N);
S:=0;
For i:=1 to N do begin                  {цикл суммирования}
a:=Sqr(2*i-1); S:=S+a end;
Sn:=N*(4*N*N-1) div 3;
Writeln('Конечная сумма S=',S:-10:2);
Writeln('Расчет конечной суммы по формуле Sn=',Sn:-10:2);
Writeln('Нажми Enter'); readln
End.
```

Варианты заданий

№ варианта	Сумма	Диапазон	n
1	2	3	4
1	$1+\frac{\ln 3}{1!}x+\frac{\ln^2 3}{2!}x^2+\ldots+\frac{\ln^n 3}{n!}x^n$	$0{,}1 \le x \le 1$	10
2	$\cos x+\frac{\cos 2x}{2!}+\ldots+\frac{\cos nx}{n!}$	$\frac{\pi}{5} \le x \le \frac{9\pi}{5}$	40
3	$x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	$0{,}1 \le x \le 1$	10
4	$1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}$	$1 \le x \le 2$	10
5	$1+\frac{\cos\frac{\pi}{4}}{1!}x+\ldots+\frac{\cos n\frac{\pi}{4}}{n!}x^n$	$0{,}1 \le x \le 1$	25
6	$1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}$	$0{,}1 \le x \le 1$	10
7	$1+\frac{\cos x}{1!}+\ldots+\frac{\cos nx}{n!}$	$0{,}1 \le x \le 1$	20
8	$1+3x^2+\ldots+\frac{2n+1}{n!}x^{2n}$	$0{,}1 \le x \le 1$	10
9	$1+\frac{x^2}{2!}+\ldots+\frac{x^{2n}}{(2n)!}$	$0{,}1 \le x \le 1$	10
10	$1+\frac{2x}{1!}+\ldots+\frac{(2x)^n}{n!}$	$0{,}1 \le x \le 1$	20
11	$1+2\frac{x}{2!}+\ldots+\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$	$0{,}1 \le x \le 1$	30
12	$1-\frac{3}{2!}x^2+\ldots+(-1)^n\frac{2n^2+1}{(2n)!}x^{2n}$	$0{,}1 \le x \le 1$	35
13	$-\frac{(2x)^2}{2!}+\frac{(2x)^4}{4!}+\ldots+(-1)^n\frac{(2x)^{2n}}{(2n)!}$	$0{,}1 \le x \le 1$	15
14	$-\frac{x}{1!}+\frac{x}{2!}+\ldots+(-1)^n\frac{x}{n!}$	$0{,}1 \le x \le 1$	15
15	$\frac{e^x}{2!}+\frac{e^x}{4!}+\frac{e^x}{6!}+\ldots+\frac{e^x}{2n!}$	$2 \le x \le 4$	10

1	2	3	4
16	$-x+\frac{\sqrt{x}}{2!}-\frac{\sqrt[3]{x}}{3!}+\ldots+(-1)^n\frac{\sqrt[n]{x}}{n!}$	$10 \le x \le 20$	10
17	$1+\frac{\sin^2 x}{3!}+\frac{\sin^4 x}{5!}+\ldots+\frac{\sin^{2n} x}{(2n+1)!}$	$\frac{\pi}{2} \le x \le \frac{3\pi}{2}$	10
18	$1+\frac{x^2}{3!}+\frac{x^4}{5!}+\ldots+\frac{x^{2n}}{(2n+1)!}$	$0{,}1 \le x \le 1$	10

Контрольные вопросы

1. Что такое конечные суммы?
2. Как осуществляется ввод-вывод конечных сумм?
3. Что такое факториал?
4. Как задается факториал в программе?
5. Составить блок-схему для вычисления конечных сумм.

Лабораторная работа № 10

ОДНОМЕРНЫЕ МАССИВЫ

Цель работы: приобретение навыков составления программы на языке **PASCAL** с использованием одномерных массивов и представление алгоритма решения задачи в виде блок-схемы.

Теоретические сведения

Массивами пользуются тогда, когда необходимо обработать большой набор однотипных данных.

Массив в языке **PASCAL** – это набор фиксированного числа некоторых значений, которые называются компонентами (элементами). Все компоненты должны быть одного типа, который называют типом элементов или базовым типом. Каждый элемент массива имеет свой порядковый номер.

Каждому конкретному массиву пользователь в своей программе дает имя, которое называют полной переменной. Каждый элемент массива обозначается полной переменной и в квадратных скобках указывается номер (индекс) этого элемента в массиве. Например, **a[1]** – элемент массива a, который стоит на первом месте; **b[k]** – элемент массива b, который стоит на k месте.

Если каждый элемент массива имеет только один индекс, то такие массивы называются одномерными. Количество элементов в массиве определяет его **размерность.**

Данные типа массив относятся к классу структурных данных. Такой тип объявляется в программе двумя способами:

1-й способ:
Var Имя массива : **Array [n..m] of** тип элементов;

2-й способ:
Type Имя типа=**Array [n..m] of** тип элементов;
Var Имя массива : Имя типа;

Слово **TYPE**, **ARRAY**, **OF** – служебные слова, переводятся соответственно *тип*, *ряд*, *из*. Имя типа и имя массива дает пользователь. В квадратных скобках указывается с какого числа по какое число могут быть номера (индексы) элементов данного массива, поэтому *n*, *m* должны быть обязательно целые числа (целые константы).

Пример. Дан одномерный массив, который состоит не более чем из n вещественных чисел. Элементы массива ввести с клавиатуры и найти сумму положительных элементов данного массива.

```
Program massiv;
Type mas = Array[1..50] of Real;
Var a:mas;
   s:Real;
   i,n:Integer;
Begin
   Writeln('Ввести размер массива');
   Readln(n);
   {Ввод элементов массива}
   For i:=1 to n do
   Begin
   Writeln('Ввести a[',i,']');
   Readln(a[i]);
   End;
{Вывод элементов массива на экран: все элементы в одну строку}
Writeln('Массив a:');
For i:=1 to n do Write(a[i]:6:2);
Writeln;
{Вычисление суммы}
s:=0;
For i:=1 to n do if a[i]>0 Then s:=s+a[i];
{вывод ответа}
Writeln('Сумма=',s:8:2);
End.
```

Варианты заданий

1. Найти наименьший из положительных элементов массива ($X1, X2, \ldots, X14$).

2. Записать в массив Y десять первых положительных элементов массива ($X1, X2, \ldots, X20$).

3. Вычислить сумму элементов массива ($A1, A2, \ldots, A12$), стоящих на четных местах.

4. Подсчитать количество положительных и отрицательных элементов в массиве ($A1, A2, \ldots, A25$). Нулевые элементы не считать.

5. Для целочисленного массива ($B1$, $B2$, ..., $B10$) определить, является ли сумма его элементов четным числом, и вывести на печать «Да» или «Нет».

6. Записать +1 вместо максимального элемента массива ($X1$, $X2$, ..., $X12$), а –1 вместо минимального.

7. Переписать положительные элементы массива ($X1$, $X2$, ..., $X30$) подряд в массив Y.

8. Определить разность между наибольшим и наименьшим элементами массива ($Y1$, $Y2$, ..., $Y20$).

9. Все элементы массива ($A1$, $A2$, ..., $A25$) уменьшить на величину среднего арифметического этих элементов.

10. Массив ($X1$, $X2$, ..., $X30$) разбить на два массива: массив положительных элементов и массив отрицательных элементов. Нуль относить к положительным элементам.

11. Переписать элементы массива ($X1$, $X2$, ..., $X20$) в обратном порядке.

12. Подсчитать сумму элементов массива ($A1$, $A2$, ..., $A22$), стоящих на четных местах, и сумму элементов того же массива, стоящих на нечетных местах.

13. Вывести на печать номера элементов ($Y1$, $Y2$, ..., $Y15$), удовлетворяющих условию $0 < Yi < 1$.

14. В массиве ($A1$, $A2$, ..., $A25$) поменять местами минимальный и максимальный элементы.

15. Дан массив ($D1$, $D2$, ..., $D45$). Найти произведение элементов с четными номерами и сумму с нечетными номерами.

16. В массиве ($F1$, $F2$, ..., $F100$) найти сумму элементов с номерами, кратными трем.

17. В массиве ($A1$, $A2$, ..., $A50$) найти количество элементов, равных X.

18. Из вектора ($A1$, $A2$, ..., $A150$) получить вектор ($B1$, $B2$, ..., $B50$), очередной компонент которого равен среднему арифметическому очередной тройки компонентов вектора A.

Контрольные вопросы

1. Формат записи одномерного массива.
2. Каким образом происходит обращение к элементам массива?
3. Какие существуют типы элементов?
4. Как осуществляется ввод-вывод элементов массива?
5. Составить блок-схему для одномерного массива.

Лабораторная работа № 11

ДВУМЕРНЫЕ МАССИВЫ

Цель работы: приобретение навыков составления программы на языке **PASCAL** с использованием двумерных массивов и представление алгоритма решения задачи в виде блок-схемы.

Теоретические сведения

После объявления массива каждый его элемент можно обработать, указав идентификатор (имя) массива и индекс элемента в квадратных скобках.

При работе с двумерным массивом указываются два индекса, с *n*-мерным массивом – *n* индексов. Например, запись **MatrU[4,4]** делает доступным для обработки значения элемента, находящегося в четвертой строке четвертого столбца массива **MatrU**.

Рассмотрим типичные ситуации, возникающие при работе с данными типа **ARRAY**.

```
VAR
  Matr:array[1..4,1..4] of integer;
```

Двумерные массивы обычно используются для представления матриц.

Для описания массива можно использовать предварительно определенные константы:

```
CONST
  G1=4; G2=6;
VAR
  MasY:array[1..G1,1..G2] of real;
```

Элементы массива располагаются в памяти последовательно. Элементы с меньшими значениями индекса хранятся в более низких адресах памяти. Многомерные массивы располагаются таким образом, что самый правый индекс возрастает самым первым. Например, если имеется массив

```
A:array[1..5,1..5] of integer;
```

то в памяти элементы массива будут размещены по возрастанию адресов:

```
A[1,1]
A[1,2]
...
A[1,5]
A[2,1]
A[2,2]
...
A[5,5]
```

Контроль правильности значений индексов массива может проводиться с помощью директивы компилятора R. По умолчанию директива R находится в пассивном состоянии {$R-}. Перевод в активное состояние вызывает проверку всех индексных выражений на соответствие их значений диапазону типа индекса.

Инициализация массива заключается в присваивании каждому элементу массива одного и того же значения, соответствующего базовому типу. Наиболее эффективно эта операция выполняется с помощью оператора **FOR**:

```
FOR I:=1 TO 10 DO
  FOR J:=1 TO 15 DO
    B[I,J]:=0;
```

Паскаль не имеет средств ввода-вывода элементов массива сразу, поэтому ввод и вывод значений производятся поэлементно.

Значения элементам массива можно присвоить с помощью оператора присваивания, как показано в примере инициализации, однако чаще всего они вводятся с помощью оператора **READ** или **READLN** с использованием оператора организации цикла **FOR**:

```
FOR I:= 1 TO 10 DO
  FOR J:= 1 TO 15 DO
    READLN (B[I,J]);
```

В связи с тем, что использовался оператор **READLN**, каждое значение будет вводиться с новой строки. Можно ввести и значения отдельных элементов, а не всего массива. Так, операторами

```
READ(B[6,9]);
```

вводится значение элемента, расположенного в шестой строке девятого столбца матрицы B.

Вывод значений элементов массива выполняется аналогичным образом, но используются операторы **WRITE** или **WRITELN**:

```
FOR I:=1 TO 10 DO
FOR J:=1 TO 15 DO
WRITELN (B[I,J]); {вывод значений массива B}
```

Варианты заданий

1. Из матрицы X(3x5) построить матрицу Y, поменяв местами строки и столбцы.

2. Дана матрица C(3x6). Найти номер строки, имеющей максимальную сумму элементов.

3. Дана матрица Q(3x3). Построить вектор ($A1$, $A2$, ..., $A9$), координатами которого являются средние арифметические элементов столбцов матрицы Q.

4. Дана квадратная матрица X(5x5). Выдать на печать элементы главной диагонали и найти сумму квадратов этих элементов.

5. Дана матрица Y(4x5). Найти минимальный элемент матрицы, а также номера строки и столбца, на пересечении которых он находится.

6. В матрице X(3x3) поменять местами строку, содержащую максимальный элемент массива, со строкой, содержащей минимальный элемент.

7. Найти разность между произведением элементов главной диагонали матрицы A(5x5) и произведением элементов побочной диагонали этой матрицы.

8. Подсчитать сумму элементов под главной диагональю квадратной матрицы X(5x5).

9. Найти минимальный элемент в каждом столбце матрицы B(4x6). Выдать на печать значения минимальных элементов с указанием номеров строк, в которых они находятся.

10. Найти сумму элементов столбца и строки матрицы D(3x5), на пересечении которых находится минимальный элемент матрицы.

11. В матрице B(5x5) найти сумму квадратов элементов, расположенных выше главной диагонали.

12. Переменной q присвоить значение наибольшего из элементов матрицы A(5x5), расположенных на главной диагонали и выше ее.

13. Найти сумму всех положительных элементов матрицы A(4x5).

14. Вывести на печать элементы главной диагонали матрицы C(5x5).

15. Найти максимальный элемент матрицы X(4x4).

16. Найти минимальный положительный элемент матрицы X(5x4).

17. Среди элементов матрицы A(4x5) найти три наибольших.

18. Просматривая матрицу B(3x5) построчно, вывести на печать первые 3 отрицательных элемента.

Контрольные вопросы

1. Формат записи двумерного массива.
2. Каким образом происходит обращение к элементам массива?
3. Какие существуют типы элементов?
4. Как осуществляется ввод-вывод элементов массива?
5. Составить блок-схему для двумерного массива.

Лабораторная работа № 12

ПОДПРОГРАММЫ

Цель работы: ознакомление с приемами программирования функций и процедур, способами обращения к ним из основной программы и представление алгоритма решения задачи в виде блок-схемы.

Теоретические сведения

Вспомогательные алгоритмы. В некоторых задачах одни и те же действия необходимо выполнять несколько раз в разных местах программы. Если записать весь текст программы, то придется один и тот же фрагмент программы писать несколько раз. В таких случаях действия, которые повторяются в разных местах программы, лучше оформить в виде отдельного алгоритма, который называется *вспомогательным*.

Вспомогательные алгоритмы в языках программирования оформляются в виде подпрограмм. В языке Паскаль подпрограммы бывают двух видов: ***процедуры пользователя*** и ***функции пользователя.***

Процедуры пользователя. Процедура пользователя имеет ту же структуру, что и любая программа языка Паскаль, другими словами, процедура пользователя состоит:

1) из заголовка;
2) объявления данных;
3) исполняемой части.

Существенные отличия процедуры и вообще программы в заголовке. Заголовок процедуры пользователя начинается служебным словом ***Procedure***, после него идет имя процедуры (которое дает пользователь), далее в круглых скобках указываются формальные параметры и их типы. Формальные параметры заголовка делятся на входные и выходные. Входные формальные параметры процедуры пользователя – это те параметры, которые получают свое значение из основной программы. Выходные формальные параметры свое значение передают в основную программу. Входные параметры

записываются сразу после открытой круглой скобки в заголовке процедуры. Выходные формальные параметры записываются после служебного слова ***Var*** в заголовке процедуры, затем круглая скобка закрывается.

Формат заголовка:

Procedure *имя процедуры (входные формальные параметры: их типы;* ***Var*** *выходные формальные параметры: их типы);*

Для обращения к процедуре в основной программе записывается процедурный оператор, который состоит из имени процедуры, и в круглых скобках перечисляются фактические параметры через запятую.

Между формальными и фактическими параметрами должно быть взаимно однозначное соответствие. Это значит, что:

1. Должно быть одинаковое количество формальных параметров в заголовке процедуры и фактических параметров в процедурном операторе.
2. Должен соблюдаться порядок следования фактических параметров в процедурном операторе и формальных параметров в заголовке процедуры.
3. Типы соответственных параметров должны быть одинаковыми.

Пример 1. Даны 3 целых числа *a*, *b*, *c*. Найти наибольший общий делитель этих чисел.

В математике имеется следующее соотношение НОД(НОД(a,b),c). Для двух целых чисел НОД находится по алгоритму Евклида. Оформим его в виде процедуры.

```
Procedure NOD (x,y:Integer; Var s:Integer);
Begin
While x<>y do If x>y Then x:= x-y Else y := y-x;
S:=x;
End;
```

В этой процедуре *x*, *y* – это два числа, для которых находим НОД. Сам НОД обозначили *s*. Таким образом *x*, *y* – это входные параметры процедуры, которые будут передаваться из основной программы в процедуру, а параметр *s* – выходной параметр, который передается из процедуры в основную программу.

Основная программа примет вид

```
Program aa;
Var a,b,c, r,t:Integer;
{Процедура пользователя}
Procedure NOD (x,y:Integer; Var s:Integer);
{x, y – входные формальные параметры}
{s – выходной формальный параметр}
{в процедуре других параметров нет, поэтому локальных параметров нет, поэтому и раздела Var в процедуре нет}
Begin
While x<>y do If x>y Then x:= x-y Else y:= y-x;
S:=x;
End;
{Основная программа}
Begin
Writeln('');
Readln(a,b,c);
NOD(a,b,r); {a,b – фактические параметры, их значения передаются переменным x, y в процедуре, r – тоже фактический параметр, но он свое значение получает из процедуры из переменной s}
NOD(r,c,t); {r,c – фактические параметры, их значения передаются переменным x, y в процедуре, t – фактический параметр, свое значение он получает из процедуры из переменной s}
Writeln('NOD(',a, ',', b, ',', c, ')=',t);
Readln;
End.
```

Входных и выходных параметров в процедуре может быть сколько угодно и они могут быть разных типов.

Функции пользователя. Функции пользователя – это другой вид подпрограмм. Функции пользователя очень похожи на процедуры пользователя, но имеют свои особенности:

1. В теле функции обязательно должен присутствовать оператор присваивания, в левой части которого стоит имя функции.

2. Обращение к функции происходит не отдельным оператором, а функция включается в состав выражений в основной программе (как обычная математическая функция).

3. Из функции пользователя можно передать только одно значение в основную программу.

Функция пользователя, как и процедура пользователя, записывается в разделе объявлений основной программы. Не имеет значения, что писать раньше процедуру пользователя или функцию, если их в программе несколько.

Формат объявления функции следующий:

Function Имя функции (Список получаемых параметров : Их типы): Тип значения функции;
Begin
Тело функции (любые операторы языка Паскаль)
End;

Обращение к функции происходит по имени функции, и в круглых скобках записываются формальные параметры, которые свое значение получают из основной программы, т. е. входные формальные параметры; указываются типы этих параметров.

Пример 2. Найти количество размещений из n элементов по m.

Решение. В математике известна формула, по которой вычисляется количество размещений $a = \frac{n!}{(n-m)!}$. Вычисление факториала необходимо произвести дважды, поэтому его вычисление лучше сделать с помощью функции пользователя.

```
Program Factorial;
Var n,m:Integer; a:Real;
{Функция пользователя}
Function Fact(k:Integer):Integer;
Var p,i:Integer;
Begin
p:=1; For i:=1 to k do p:=p*i; Fact:=p;
End;
{Основная программа}
```

```
Begin
 Writeln('Ввести значения n и m'); Readln(n,m);
 a:=Fact(n)/Fact(n-m);
 Writeln('Число размещений=',a:5:0);
 Readln;
End.
```

Замечание. Любую функцию пользователя можно записать в виде процедуры пользователя, а наоборот – не всегда.

Варианты заданий

Вычисления произвести в подпрограмме. В качестве параметров-значений описать массив. В теле основной программы осуществить ввод массива и далее обратиться к подпрограмме.

1. Найти наименьший из положительных элементов массива ($X1, X2, \ldots, X14$).
2. Записать в массив Y десять первых положительных элементов массива ($X1, X2, \ldots, X20$).
3. Вычислить сумму элементов массива ($A1, A2, \ldots, A12$), стоящих на четных местах.
4. Подсчитать количество положительных и отрицательных элементов в массиве ($A1, A2, \ldots, A25$). Нулевые элементы не считать.
5. Для целочисленного массива ($B1, B2, \ldots, B10$) определить, является ли сумма его элементов четным числом, и вывести на печать «Да» или «Нет».
6. Записать +1 вместо максимального элемента массива ($X1, X2, \ldots, X12$), а –1 вместо минимального.
7. Переписать положительные элементы массива ($X1, X2, \ldots, X30$) подряд в массив Y.
8. Определить разность между наибольшим и наименьшим элементами массива ($Y1, Y2, \ldots, Y20$).
9. Все элементы массива ($A1, A2, \ldots, A25$) уменьшить на величину среднего арифметического этих элементов.
10. Массив ($X1, X2, \ldots, X30$) разбить на два массива: массив положительных элементов и массив отрицательных элементов. Нуль относить к положительным элементам.

11. Переписать элементы массива ($X1$, $X2$, ..., $X20$) в обратном порядке.

12. Подсчитать сумму элементов массива ($A1$, $A2$, ..., $A22$), стоящих на четных местах, и сумму элементов того же массива, стоящих на нечетных местах.

13. Вывести на печать номера элементов ($Y1$, $Y2$,...,$Y15$), удовлетворяющих условию $0 < Yi < 1$.

14. В массиве ($A1$, $A2$, ..., $A25$) поменять местами минимальный и максимальный элементы.

15. Дан массив ($D1$, $D2$, ..., $D45$). Найти произведение элементов с четными номерами и сумму с нечетными номерами.

16. В массиве ($F1$, $F2$, ..., $F100$) найти сумму элементов с номерами, кратными трем.

17. В массиве ($A1$, $A2$, ..., $A50$) найти количество элементов, равных X.

18. Из вектора ($A1$, $A2$, ..., $A150$) получить вектор ($B1$, $B2$, ..., $B50$), очередной компонент которого равен среднему арифметическому очередной тройки компонентов вектора A.

Контрольные вопросы

1. Способы описания функций и процедур.
2. Как задаются параметры-переменные и параметры-значения?
3. Как осуществляется вызов функции и процедуры?
4. В чем отличие функции от процедуры?
5. Составить блок-схему с использованием функции и процедуры.

Лабораторная работа № 13

ЗАПИСИ

Цель работы: изучение записей при решении задач на языке **PASCAL** и представление алгоритма решения задачи в виде блок-схемы.

Теоретические сведения

ЗАПИСЬ – это структурированный тип данных, состоящий из фиксированного числа компонентов одного или нескольких типов. Определение типа записи начинается идентификатором **RECORD** и заканчивается зарезервированным словом **END**. Между ними заключен список компонентов, называемых полями, с указанием идентификаторов полей и типа каждого поля.

Формат:

```
TYPE
<имя типа> = RECORD
            <идентификатор поля>:<тип компонента>;
            ...
            <идентификатор поля>:<тип компонента>
        END;
VAR
<идентификатор,...> : <имя типа>;
```

Пример 1.

```
TYPE
   Car = RECORD
      Number:integer; {номер}
      Marka:string[20]; {марка автомобиля}
      FIO:string[40]; {фамилия, инициалы владельца}
      Address:string[60] {адрес владельца}
    END;
VAR
M, V:Car;
```

В данном примере запись **Car** содержит четыре компонента: номер, название марки машины, фамилию владельца и его адрес. Доступ к полям записи осуществляется через переменную типа запись. В нашем случае это переменные **M** и **V** типа **Car**.

Идентификатор поля должен быть уникален только в пределах записи, однако во избежание ошибок лучше делать его уникальным в пределах всей программы. Объем памяти, необходимый для записи, складывается из длин полей.

Значения полей записи могут быть использованы в выражениях. Имена отдельных полей не применяются по аналогии с идентификаторами переменных, поскольку может быть несколько записей одинакового типа. Обращение к значению поля осуществляется с помощью идентификатора переменной и идентификатора поля, разделенных точкой. Такая комбинация называется составным именем. Например, чтобы получить доступ к полям записи **Car**, надо записать:

```
M.Number, M.Marka, M.FIO, M.Address
```

Составное имя можно использовать везде, где допустимо применение типа поля. Для присваивания полям значений используется оператор присваивания.

Пример 2.

```
M.Number:=1678;
M.Marka:='ГАЗ-24';
M.FIO:='Орлов А.Г. ';
M.Address:='ул. Пушкина 12 - 31';
```

Составные имена можно использовать, в частности, в операторах ввода-вывода:

```
Read(M.Number, M.Marka, M.FIO, M.Address);
Write(M.Number:4, M.Marka:7, M.FIO:12, M.Address:25);
```

Допускается применение оператора присваивания и к записям в целом, если они имеют один и тот же тип. Например,

```
V:=M;
```

После выполнения этого оператора значения полей записи V станут равны значениям соответствующих полей записи M.

В ряде задач удобно пользоваться массивами из записей. Их можно описать следующим образом:

```
TYPE
  Person = RECORD
     FIO:string[20];
     Age:1 .. 99;
     Prof:string[30]
  END;
VAR
  List:array[1..50] of Person;
```

Пример 3.

Используя тип-запись, разработать базу данных «Телефонный справочник» (фамилия, имя, телефон). Вывести на экран данные по владельцам телефонов, имеющим введенную с клавиатуры фамилию. Предусмотреть сохранение двух таблиц в файлах на диске.

```
Program pr_12;
   uses crt;
   type person=record
     sname,name:string[14];
     tel:longint end;
   var rbook:array[1..20] of person;
     f1,f2:text; i,n:integer; st1:string;
   BEGIN clrscr; Assign(f1,'p1.pas'); Rewrite(f1);
     writeln('Введите количество записей'); readln(n);
     for i:=1 to n do with rbook[i] do begin
     WRITELN('Фамилия '); readln(sname);
     writeln(f1,sname);
     WRITELN('имя'); readln(name); writeln(f1,name);
     WRITELN('Телефон');readln(tel); writeln(f1,tel); end;
     Assign(f2,'p2.pas'); rewrite(f2);
     Writeln('Введите фамилию для поиска');
Readln(st1); Writeln('Фамилия Имя Телефон');
     Writeln(f2,'Фамилия Имя Телефон');
```

```
    for i:=1 to n do begin With rbook[i] do
    if sname=st1 then begin
Writeln(sname:12,name:10,tel:10);
    Writeln(f2,sname:12,name:10,tel:10);
    end; end;close(f1); Close(f2); readln; END.
```

Варианты заданий

1. Составить программу формирования телефонного справочника и выдачи номера телефона по указанной фамилии. Число абонентов принять равным 8. Назначение полей записи: фамилия, имя, отчество; номер телефона.

2. Составить программу формирования справочных сведений о городах с последующим получением необходимой информации о нужном городе. Число городов принять равным 8. Назначение полей записи: город, статус, число жителей, занимаемая площадь.

3. Составить программу формирования списка студентов разных групп и выдачи на выводное устройство списка студентов группы. Взять число студентов равное 10. Назначение записи: группа; ФИО, год поступления; год рождения.

4. Составить программу формирования сводной ведомости участников соревнований и выдачи на выходное устройство фамилий участников, занявших 1, 2 и 3-е места. Число участников – 9. Назначение полей записи: ФИО, пол; год рождения; место.

5. Составить программу формирования справочника технического обслуживания автомобилей и выдачи на выходное устройство названия станции и ее адреса, обслуживающей данный автомобиль. Число станций – 10. Назначение полей: название станции; адрес; марка автомобиля.

6. Составить программу формирования справочника деталей на складах и выдачи сообщения о наличии данной детали. Назначение полей записи: код детали; номер стеллажа; название склада. Число деталей – 10.

7. Разработать программу формирования файла, содержащего сведения о студентах. Каждый элемент этого файла должен содержать следующие данные: номер группы; номер в группе по списку; ФИО; год рождения; оценки за последнюю сессию.

8. Разработать программу формирования и корректировки (отдельных элементов) файла, в котором хранятся сведения о товарах, находящихся на складе. Каждый элемент этого файла должен содержать следующую информацию: наименование товара; объем партии; дату поступления на склад; стоимость единицы товара.

9. Разработать программу создания и корректировки (добавление новых элементов) файла, содержащего сведения о книгах, находящихся в читальном зале библиотеки. Каждый элемент этого файла должен содержать следующую информацию: ФИО автора; название книги; наименование издательства; год издания; количество страниц.

10. Задана ведомость абитуриентов, сдавших вступительные экзамены в институт. В каждой строке данной ведомости записана фамилия абитуриента, специальность, на которую он поступает, и полученные им оценки по отдельным дисциплинам (например, физике, математике, литературе). Написать программу для хранения указанной информации. Программа должна предусматривать ввод, корректировку, дополнение ведомости, а также формирование выходных текстовых файлов по каждой специальности, содержащих фамилии и инициалы абитуриентов и суммарный балл для каждого из них.

11. В справочной аэропорта хранится информация о вылете самолетов на следующие сутки. Для каждого рейса указаны номер рейса, тип самолета, пункт назначения, время вылета. Имеются справочники по расстояниям между всеми возможными пунктами назначения, по расходу горючего на тысячу километров для каждого самолета. Написать программу выдачи справочной информации о рейсе по пункту назначения.

12. У администратора железнодорожных касс хранится информация о свободных местах в поездах по всем направлениям на ближайшую неделю. Данная информация представлена в следующем виде: дата выезда, номер рейса, конечный пункт назначения, время отправления, число свободных купейных мест, число свободных плацкартных мест. Программа должна позволять корректировать записи, выдавать информацию об имеющихся местах по каждому рейсу и каждому типу мест (купейные или плацкартные).

13. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизор, радиоприемник и т. п.), марку изделия, дату приемки в ремонт, состояние готовности заказа

(выполнен, не выполнен). Разработать программу для ведения архива квитанций на персональной ЭВМ. В архиве должны храниться квитанции за последний квартал. Необходимо выдавать на основании анализа архива информацию начальнику ателье о числе и характере заказов на текущие сутки и объеме выполненных услуг за текущий квартал.

14. Написать программу для хранения информации об успеваемости студентов. Необходимо хранить номер группы, ФИО студента, оценки за последнюю сессию, корректировать указанную информацию и распечатывать списки студентов по группам с указанием среднего балла каждого за последнюю сессию. В каждой группе фамилии студентов в распечатке размещать в порядке убывания среднего балла.

15. В магазине имеется список лиц, записавшихся на покупку мебельного гарнитура. Каждая запись этого списка содержит порядковый номер, ФИО, домашний адрес покупателя и дату постановки на учет. Написать программу для хранения этого списка. Программа должна по запросу пользователя выдавать информацию об общем числе записавшихся в каждом микрорайоне города (определять по названию улицы), упорядочить по дате постановки на учет.

16. В больнице имеется общий список больных, каждый из которых характеризуется записью

```
Type
ZAP=Record
F10:string [25];
NP:integer ;
SEX:string [3];
DGN:string [20];
end;
```

где F10 – ФИО больного, NP – номер палаты, SEX – пол (мужской или женский), DGN – диагноз. Написать программу для хранения и корректировки списка больных. Программа должна позволять выводить данные о новых больных, размещать их по палатам с учетом диагноза, выводить по запросам пользователя информацию о больных по палатам, находить номер палаты, в которой лежит нужный больной.

17. В бюро по занятости населения (трудовой бирже) ведется список вакантных рабочих мест на предприятиях города. Каждая запись такого списка содержит следующую информацию: наименование организации, местоположение организации (в километрах от центра города), наименование должности, требуемая квалификация (разряд или образование), требуемый стаж работы по специальности, заработная плата в месяц, наличие социального страхования (да или нет), продолжительность ежегодного оплачиваемого отпуска. Клиент бюро вводит информацию о своей квалификации и требованиях (например, максимальная удаленность от центра города). Написать программу, которая бы позволяла хранить информацию указанной структуры, вносить сведения о вновь появившихся свободных местах, удалять информацию об уже занятых местах, распечатывать для каждого клиента список рабочих мест в соответствии с его требованиями.

Контрольные вопросы

1. Что такое запись?
2. Как объявляется запись?
3. Как объявить массив записей?
4. Как происходит обращение к компонентам записи?
5. Как осуществляется ввод-вывод записи?

Лабораторная работа № 14

ФАЙЛЫ

Цель работы: изучение особенностей языка Паскаль, связанных с обработкой данных, объединенных в файлы.

Теоретические сведения

ФАЙЛ – это поименованная область памяти на внешнем носителе, предназначенная для хранения данных. В большинстве случаев файлы состоят из текстовых строк или записей и размещаются на гибких или жестких дисках.

Для описания файла используется словосочетание **file of**. Для доступа к файлу описывается специальная файловая переменная (обозначим ее как **F**). Если файл состоит из записей, дополнительно описывается переменная для доступа к полям записи (обозначим ее как **R**).

Формат:

```
TYPE
  <имя типа> = <тип компонента>;
VAR
  <F> : file of <имя типа>;
  <R> : <имя типа>;
```

В **PASCAL** существуют три класса файлов: текстовые, типизированные и нетипизированные. Файловая система на **PASCAL** наиболее полно использует возможности операционной системы **DOS** по передаче данных. Каждому файлу в языке ставится в соответствие файловая переменная определенного типа, поэтому перед началом работы с файлом необходимо установить данное соответствие. Для этого в языке используется процедура

Assign (VAR F; Name: string);

где **F** – переменная любого файлового типа, а строковое выражение **Name** содержит полное имя файла, удовлетворяющее требованиям операционной системы. Обобщенный вид имени файла выглядит следующим образом:

Диск:\ИмяПодкаталога\...\ИмяПодкаталога\ИмяФайла

Текстовый файл можно рассматривать как последовательность символов, разбитую на строки длиной от 0 до 256 символов. Для описания используется стандартный тип Text:

```
VAR
F: text; {F – файловая переменная}
```

У текстовых файлов своя специфика. Специальные расширения стандартных процедур чтения **Read** и записи **Write** разрешают работать со значениями несимвольного типа. Другими словами, последовательность символов автоматически преобразуется к значению того типа переменной, который используется в файловых операциях.

Вызов **Read(F, Ww)**, где **Ww** – переменная типа word, осуществляет чтение из файла **F** последовательности цифр, которая затем интерпретируется в число, значение которого и будет присвоено переменной **Ww**. В случае, если вместо последовательности цифр идет любая другая последовательность символов, использование такого оператора приводит к ошибке выполнения программы.

Открытие текстового файла можно произвести стандартным способом:

поставить в соответствие файловой переменной имя файла (процедура **Assign**),

открыть новый текстовый файл (процедура **Rewrite**);

либо поставить в соответствие файловой переменной имя файла (процедура **Assign**),

открыть уже существующий файл (процедура **Reset**).

Текстовый файл в силу своей специфики во время работы допускает какой-либо один вид операции: чтение или запись. В связи с этим для работы с текстовыми файлами появляется еще одна процедура открытия файла:

Append(VAR F:text);

Эта процедура открывает уже существующий файл и позиционирует указатель обработки на конец файла. После такого открытия текстовый файл можно только дополнять информацией, начиная с конца строки. Ограничения, накладываемые на процедуру **Append**, такие же, как у процедур **Reset** и **Rewrite**.

Для обработки текстовых файлов используются процедуры **Read** и **Write**, обеспечивающие соответственно чтение и запись одной и более строк в текстовый файл. Использование специальных разделителей строк файла позволило ввести в состав языковых средств процедуры: **Readln**, обеспечивающую те же действия, что и **Read**, и дополнительно – чтение до маркера конца строки и переход к новой строке; **Writeln**, обеспечивающую запись всех величин с обязательной установкой маркера конца строки в файл. Общий вид представления процедур следующий:

Readln(VAR F:text; V1 [,V2,...Vn]);
Writeln(VAR F:text; V1 [,V2,...Vn]);

где **V1...Vn** переменные разных типов. Процедура **Read** обеспечивает ввод данных общим потоком из одной строки, а **Readln** приводит к обязательному переходу к следующей строке текстового файла, т. е. ввод данных осуществляется из различных строк. Все вышесказанное в равной мере относится к операциям записи с помощью процедур **Write** и **Writeln**.

При организации операций ввода-вывода используются специальные языковые средства в виде функций **Eoln**, **Eof**, **SeekEoln**, **SeekEof**.

Функция **Eoln(VAR F:text)** возвращает булевское значение **True**, если текущая файловая позиция находится на маркере конца строки или вызов **Eof(F)** возвратил значение **True**. Во всех других случаях значение функции будет **False**.

Функция **Eof(VAR F:text)** возвращает булевское значение **True**, если указатель конца файла находится сразу за последним компонентом, и **False** – в противном случае.

Функция **SeekEoln(VAR F:text)** возвращает булевское значение **True** при достижении маркера конца строки, причем указатель файла пропускает все пробелы и знаки табуляции, предшествующие маркеру. В противном случае функция возвращает значение **False**.

Функция **SeekEof(VAR F:text)** возвращает значение **True**, если указатель файла находится на маркере конца файла. Эта функция также пропускает все пробелы и знаки табуляции, предшествующие

маркеру, и выполняет автоматический пропуск маркера конца строки. Характерным примером использования этих функций может служить чтение числовых величин из текстового файла, когда необходимо пропустить обработку разделяющих эти числа пробелов или знаков табуляции.

Варианты заданий

Создать текстовый файл исходных данных (элементы массива) с расширением .TXT. Создать программу, в которой элементы массива считываются из файла исходных данных, выполнить требуемые преобразования и результат вывести в файл с расширением .REZ.

1. Из матрицы $X(3x5)$ построить матрицу Y, поменяв местами строки и столбцы.

2. Дана матрица $C(3x6)$. Найти номер строки, имеющей максимальную сумму элементов.

3. Дана матрица $Q(3x3)$. Построить вектор $(A1, A2, \ldots, A9)$, координатами которого являются средние арифметические элементов столбцов матрицы Q.

4. Дана квадратная матрица $X(5x5)$. Выдать на печать элементы главной диагонали и найти сумму квадратов этих элементов.

5. Дана матрица $Y(4x5)$. Найти минимальный элемент матрицы, а также номера строки и столбца, на пересечении которых он находится.

6. В матрице $X(3x3)$ поменять местами строку, содержащую максимальный элемент массива, со строкой, содержащей минимальный элемент.

7. Найти разность между произведением элементов главной диагонали матрицы $A(5x5)$ и произведением элементов побочной диагонали этой матрицы.

8. Подсчитать сумму элементов под главной диагональю квадратной матрицы $X(5x5)$.

9. Найти минимальный элемент в каждом столбце матрицы $B(4x6)$. Выдать на печать значения минимальных элементов с указанием номеров строк, в которых они находятся.

10. Найти сумму элементов столбца и строки матрицы $D(3x5)$, на пересечении которых находится минимальный элемент матрицы.

11. В матрице $B(5x5)$ найти сумму квадратов элементов, расположенных выше главной диагонали.

12. Переменной q присвоить значение наибольшего из элементов матрицы $A(5x5)$, расположенных на главной диагонали и выше ее.

13. Найти сумму всех положительных элементов матрицы $A(4x5)$.

14. Вывести на печать элементы главной диагонали матрицы $C(5x5)$.

15. Найти максимальный элемент матрицы $X(4x4)$.

16. Найти минимальный положительный элемент матрицы $X(5x4)$.

17. Среди элементов матрицы $A(4x5)$ найти три наибольших.

18. Просматривая матрицу $B(3x5)$ построчно, вывести на печать первые 3 отрицательных элемента.

Контрольные вопросы

1. Типы файлов.
2. Формат записи текстового и типизированного файлов.
3. Функции для работы с файлами.
4. Процедуры для работы с файлами.

Литература

1. Использование Microsoft Office. Специальное издание / К. Кенни [и др.]. – Киев: Диалектика, 1995. – 480 с.

2. Микляев, А. Настольная книга пользователя IBM PC. – Изд. 2-е, перераб. и доп. – М.: Солон, 1998. – 608 с.

3. Визе, Манс. Word 6.0 для Windows (русская версия) / Пер. с нем. – М.: БИНОМ. – 208 с.

4. Фаненштих, Клаус, Хаселир, Райнер. Текстовый процессор Word 6.0 для Windows: практическое пособие / Пер. с нем. – Изд. 2-е, исправ. и доп. – М.: ЭКОМ, 1995. – 352 с.

5. Долголаптев, В.Г. Работа в Excel 7.0 для Windows 95 на примерах. – М.: БИНОМ, 1995. – 383 с.

6. Каймин, В.А. Информатика: учебник. – М.: ИНФРА-М, 2000. – 232 с. – (Серия «Высшее образование»).

7. Рудикова, Л.В., Соболевский, С.Л., Шишканов, М.М. Microsoft Excel в науке и бизнесе. – Минск: ИООО «Право и экономика», 2003. – 247 с.

8. Паскаль для персональных компьютеров: справ. пособие / Ю.С. Бородич [и др]. – Минск: Выш. шк.; БФ ГИТМП «НИКА», 1991. – 365 с.

9. Фигорнов, В.Э. IBM PC для пользователя. – 6-е изд., перераб. и доп. – М.: Инфра-М, 1995. – 432 с.

10. Фаронов, В.В. Турбо Паскаль 7.0. Практика программирования. – М., 1999. – 475 с.

Учебное издание

ИНФОРМАТИКА

Лабораторный практикум
для студентов специальности
1-43 01 04 «Тепловые электрические станции»

В 2 частях

Часть 1

Составители:
ТАРАСЕВИЧ Леонид Александрович
ПРОНКЕВИЧ Елена Васильевна
ПОПОВА Юлия Борисовна

Редактор Н.В. Артюшевская
Технический редактор О.В. Дубовик
Компьютерная верстка О.В. Дубовик

Подписано в печать 11.10.2007.
Формат 60×84 1/16. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 5,35. Уч.-изд. л. 4,18. Тираж 100. Заказ 708.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0131627 от 01.04.2004.
220013, Минск, проспект Независимости, 65.