Министерство образования и науки Украины

Севастопольский национальный технический университет

Методические указания
к выполнению лабораторных работ по дисциплине
"ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЕ"
ЧАСТЬ 1. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ
для студентов дневной формы обучения направления 7.100201-"Кораблестроение и океанотехника"

Севастополь
2005

УДК 655.3.06

Методические указания к выполнению лабораторных работ по дисциплине “Основы информатики и программирование” Часть 1. Основы алгоритмизации и программирования. Для студентов дневной формы обучения направления "Кораблестроение и океанотехника"/ Е.М. Шалимова.- Севастополь: Изд-во СевНТУ, 2005.- 68с.

Цель методических указаний – изложение основных сведений по работе в среде Turbo Pascal, описание порядка выполнения лабораторных работ и индивидуальных заданий.

Методические указания рассмотрены и утверждены на заседании кафедры кибернетики и вычислительной техники, протокол № 3 от 21.11.05.

Рецензенты:
Тертычный А.И., старший преподаватель кафедры КиВТ;
Первухина Е.Л., доктор техн. наук, профессор кафедры ТМ.

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Ответственный за выпуск: заведующий кафедрой кибернетики и вычислительной техники проф. Скатков А.В.

Содержание

1. КРАТКИЕ СВЕДЕНИЯ ОБ ОБОЛОЧКЕ СИСТЕМЫ TURBO PASCAL

1.1. Вход в систему и выход из нее

Для входа в редактор системы в командной строке DOS следует набрать:

turbo

или

turbo <имя файла> .

В последнем случае при входе в редактор загрузится файл с указанным именем. Вид окна редактора представлен на рисунке 1. Для выхода из редактора следует нажать комбинацию клавиш **Alt+X** (в любой момент, кроме счета программы).

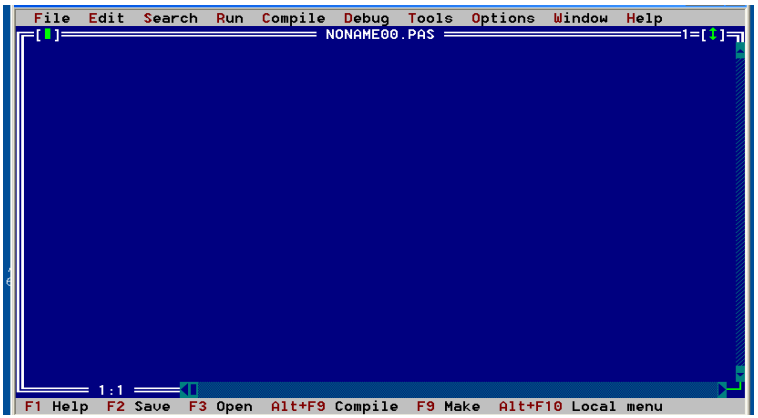


Рисунок 1- Вид окна редактора системы Turbo Pascal

1.2. Основное меню системы

Переход в основное меню (верхняя строка экрана) из редактора осуществляется нажатием клавиши **F10.** Для выбора раздела основного меню его необходимо выделить (клавишами $\rightarrow$ и $\leftarrow$) и нажать **Enter;** появится

меню с командами этого раздела. Возврат из основного меню в редактор производится нажатием клавиши **Esc.**

Для некоторых наиболее часто используемых команд введены функциональные клавиши или комбинации клавиш (F1, F2, Alt+F9 и т.п.), нажатие которых в редакторе эквивалентно вызову этих команд. Эти клавиши указываются в скобках при описании команд. Например, для выхода из системы можно просто нажать **Alt+X** вместо того, чтобы в разделе **File** основного меню выбирать команду **Exit.**

1.3. Общие сведения по работе с меню

Меню - перечень разделов, один из которых надо выбрать для продолжения работы. Один из разделов меню отмечается особым цветом, это - выделенный раздел.

Выделение нужного раздела меню осуществляется **клавишами со стрелками** или **Tab**. Выбор выделенного раздела осуществляется нажатием клавиши **Enter**; отказ от выбора и выход из меню - нажатием клавиши **Esc**.

Если за названием раздела следует многоточие (...), то выбор раздела приведет к появлению диалогового окна. Если за названием следует стрелка (>), то выбор раздела приведет в другое меню (подменю). Раздел без таких знаков указывает, что при его выборе сразу произойдет указанное действие.

Меню вида "диалоговое окно" позволяет менять его содержимое в диалоге с пользователем. Такое окно содержит элементы нескольких типов (переходы между элементами разных типов осуществляется нажатием клавиш **Tab** или **Shift+Tab**, активный элемент высвечивается особым цветом):

1) Строка ввода - используется для набора какого-то текста (например, имени файла); при вводе текста

допускается его редактирование; конец ввода – нажатие **Enter**.

2) *Список* - вертикальный перечень каких-либо объектов (например, имен файлов), из которых надо выбрать один. Выбор осуществляется клавишами ← и → с последующим нажатием **Enter**.

3) *Независимые кнопки* (перед ними - квадратные скобки) используются для установки характеристик (например, условий поиска некоторого фрагмента текста в файле), которые независимы друг от друга и потому могут быть "включены" одновременно ("включенные" кнопки помечаются символом X). Выбор осуществляется клавишами **Tab** либо пробел, либо нажатием высвеченной буквы; если кнопка "включена", то повторный выбор "выключит" ее.

4) *Зависимые кнопки* (перед ними - круглые скобки) - аналог независимых кнопок, но выбор одной кнопки исключает выбор другой. "Включенные" зависимые кнопки помечаются точкой в круглых скобках.

После заполнения диалогового окна нужной информацией необходимо выполнить "завершающее" действие - указать, что делать с этим окном. Для этого используются:

5) *Кнопки действия* (они в прямоугольниках): **OK** - успешно завершить работу с диалоговым окном; **Cancel** (или клавиша **Esc**) - отказ от всех действий в этом окне; **Help** - вызов помощи (подсказки) и т.д. Для выбора нужной кнопки используется клавиша **Tab** с последующим нажатием **Enter**.

1.4 Раздел EDIT (редактор)

В данном разделе собраны команды, позволяющие менять содержимое файла (редактировать его).

1.4.1. Окна редактора

Редактор работает с несколькими окнами (экранами) , в каждом из которых размещен текст одного из файлов. Окна нумеруются (1, 2, ...), номер окна указывается в правом верхнем углу. Окна "накладываются" друг на друга; самое "близкое" на экране окно является активным, именно с ним работают редактор и компилятор; это окно окружено двойной рамкой.

Если текст одного и того же файла находится в нескольких окнах, то изменение его в одном окне приводит к изменению его и в других окнах.

В нижнем левом углу окна указываются номера строки и столбца текста, в которых в данный момент находится курсор. Если текст в окне был изменен, то слева от этих номеров появится звездочка (*).

Alt+k - сделать активным окно с номером k;

Alt+0 - выдать перечень всех окон и имен их файлов;

Alt+F3 - закрыть активное окно и вернуться к предпоследнему активному окну;

F6 - перейти к следующему окну;

Shift+F6 - перейти к предыдущему окну;

Ctrl+F5 - изменить размер и позицию на экране активного окна (стрелки меняют позицию, а **Shift**+**стрелки**-размер; конец изменений - **Enter**, отказ от изменений – **Esc)**.

1.4.2. Основные команды редактирования

Для перемещения по тексту используются клавиши управления курсором и их комбинации с управляющими клавишами:

$\rightarrow$, $\leftarrow$, $\uparrow$, $\downarrow$ - перемещение курсора на одну позицию в соответствующем направлении;

Home, **End** - перемещение курсора в начало или конец текущей строки соответственно;

PgUp, **PgDown** - листание страниц (размером в окно) вверх или вниз;
Ctrl+PgUp, **Ctrl+PgDown** - переход на начало или конец файла.

Ввод текста возможен в одном из двух режимов: в режиме вставки или в режиме замены.

Режим вставки обеспечивает вставку символа в позицию курсора, при этом текст, находящийся справа от курсора, сдвигается на одну позицию вправо. *Режим замены* обеспечивает замену символа в позиции курсора.

Переключение между режимами вставки и замены происходит при нажатии клавиши **Ins** (в режиме замены размер курсора увеличен).

Основные клавиши редактирования:
Del - удалить символ, на который указывает курсор;
Backspace - удалить символ слева от курсора;
Enter - в позиции курсора разбить строку на две;
Ctrl+Y - удалить текущую строку.

1.4.3. Работа с блоками и карманом

Блок - участок текста, с которым можно работать как с единым целым. Действия над блоком выполняются, только если он выделен (высвечен).

Карман (буфер) - некоторая вспомогательная память, где можно хранить фрагмент текста (до следующего изменения кармана). Карман нужен для переноса блоков из одного окна в другое (для этого надо выделить блок в одном окне, и скопировать блок в карман, а затем перейти в другое окно и считать блок из кармана) или из одной позиции в другую.
Ctrl+K+B - определить (в позиции курсора) начало блока;
Ctrl+K+K - определить (в позиции курсора) конец блока;
Ctrl+K+H - выделить или снять выделение блока;

Ctrl+K+V - перенести блок на новое место (в позицию курсора) с уничтожением блока на старом месте;
Ctrl+K+C - скопировать блок на новом месте (в позиции курсора) с сохранением блока на старом месте;
Ctrl+Del - удалить блок из текста;
Shift+Del - удалить блок с записью его в карман (команда **Cut** из меню **Edit** *)*;
Ctrl+Ins - скопировать блок в карман (команда **Copy** из меню **Edit**) ;
Shift+Ins **-** вставить блок из кармана в позицию курсора активного окна (команда **Paste** из меню **Edit**).

1.5. Раздел FILE (работа с файлами)

Данный раздел содержит команды для работы с файлами.
Open (F3) - считать с диска новый файл для обработки в редакторе. По этой команде появляется диалоговое окно, в котором надо указать имя файла. Файл загрузится в новое окно редактора, если выбрана кнопка действия **Open** , либо заменит в текущем окне прежний файл, если выбрана кнопка **Replace** . После выбора файла происходит автоматический возврат в редактор. Отказ от открытия файла - **Esc** или кнопка **Cancel** .
New - создать новый (пустой, безымянный) файл в новом окне редактора.
Save (F2) - сохранить на диске файл из текущего окна под ПРЕЖНИМ именем.
Save as **-** сохранить файл под новым именем, которое надо будет указать дополнительно (система запросит его).
Save all - сохранить все файлы (из всех окон редактора).
Change dir - выбрать нужную директорию и сделать ее текущей (в дальнейшем файлы будут выбираться из этой директории). Имя директории либо указывается явно, либо

выбирается перемещением по дереву директорий, изображаемое данной командой на экране.
Exit (Alt+X) - выход из системы Turbo Pascal.

1.6. Раздел COMPILE (трансляция программы)

Раздел **COMPILE** содержит команды, с помощью которых можно управлять процессом трансляции программы.
Compile (Alt+F9)- трансляция программы из текущего окна. Оттранслированная программа либо размещается в памяти, либо записывается на диск (с расширением EXE).
Destination- у этой команды два параметра: Memory и Disk, которые указывают, где транслятор должен размещать оттранслированную программу - в памяти или на диске. Для смены параметра надо выделить эту команду и нажать **Enter**.

1.7. Раздел RUN (выполнение программы)

Раздел **RUN** содержит команды, управляющие процессом исполнения программы.
Run (Ctrl+F9) - запуск на счет программы из текущего окна. Если программа не была ранее оттранслирована, то **Run** предварительно оттранслирует ее и разместит машинную программу в памяти или на диске согласно параметру команды **Destination** из раздела **Compile**. Если программа была ранее оттранслирована, и после этого ее текст не менялся, то **Run** не будет перетранслировать ее.

На время выполнения программы экран редактора исчезает и появляется экран пользователя, на котором отображаются данные, вводимые пользователем в программу, и результаты работы программы. По окончании счета происходит возврат в редактор и снова восстанавливается экран редактора. Если надо

посмотреть экран пользователя, следует набрать **Alt+F5**; нажатие затем любой клавиши восстановит экран редактора.

1.8. Раздел HELP ("помощь", справочная информация)

Раздел **Help**(справка*)* – справочная информация по системе Turbo Pascal.Его можно вызвать (в любой момент, кроме счета) по клавише **F1**.

Сначала появится раздел **Help'**а**,** соответствующий тому разделу меню, из которого вызван **Help.** Если еще раз нажать **F1,** то на экране появится общее оглавление **Help***'*а, из которого можно попасть в любой раздел справочной системы.

Если перед обращением к справочной системе с помощью курсора выделить некоторый элемент программы, то при нажатии **F1** на экране появится информация по выделенному элементу.

2. ПОНЯТИЕ АЛГОРИТМА. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Фундаментальным понятием в области компьютерных наук является понятие ***алгоритма.*** Часто это понятие определяют так [1]:

алгоритм- это точное предписание о порядке выполнения действий из заданного фиксированного множества для решения всех задач заданного класса.

Рассмотрим подробнее ключевые слова в этой формулировке:

"точное предписание" означает, что предписание однозначно и одинаково понимается всеми исполнителями алгоритма и при одних и тех же исходных данных любой исполнитель всегда получает один и тот же результат;

"из заданного фиксированного множества" означает, что множество действий, используемых в предписании, оговорено заранее и не может меняться в ходе исполнения алгоритма;

"решение всех задач заданного класса" означает, что это предписание предназначено для решения множества задач, а не одной отдельной задачи.

Исходные данные - это значения, с которых начинается исполнение алгоритма. Множество исходных данных всегда точно определено. Оно может быть определено явно, перечислением его элементов, либо неявно, в виде системы правил, определяющих конструкцию его элементов.

Вычислительным процессом, порожденным алгоритмом, называется последовательность шагов алгоритма, пройденных при исполнении этого алгоритма.

Сложностью алгоритма называется количество действий в вычислительном процессе этого алгоритма.

Основные свойства алгоритма:

- ***массовость*** (алгоритм предназначен для решения множества задач некоторого класса, а не одной отдельной задачи);
- ***потенциальная осуществимость*** (конечность, выполнение алгоритма должно приводить к его завершению за конечное число шагов);
- ***детерминированность*** (алгоритм всегда определяет однозначно, какое действие должно быть выполнено следующим, равно как и то, какое действие должно быть выполнено первым.);
- ***однозначность*** (при одних и тех же исходных данных любой исполнитель всегда получает один и тот же результат).

Основные этапы решения любой задачи были выделены математиком Дж. Полиа еще в 1945 году [1]. Применительно к разработке программ эти этапы определяются так:

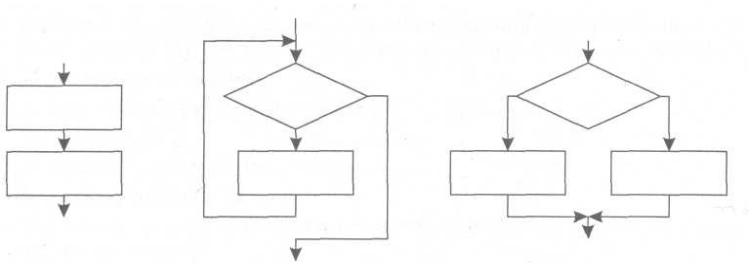
- сформулировать постановку задачи:
- построить математическую модель;
- разработать алгоритм и представить его в виде программы;
- отладить и протестировать программу;
- проанализировать результаты.

Одним из способов представления алгоритмов является структурная блок- схема (или просто блок-схема) алгоритма [2]. Структурная блок-схема - это блок-схема, которая может быть выражена как композиция элементарных блок-схем, называемых следованием, ветвлением и циклом (рисунок 2).

Следованием называется конструкция, представляющая собой последовательное выполнение двух или более операторов (простых или составных). Ветвление задает выполнение либо одного, либо другого оператора в зависимости от выполнения некоторого условия. Цикл задает многократное выполнение

оператора. В теории программирования доказано, что алгоритм и соответствующую программу для решения задачи любой сложности можно составить только из трех указанных структур.

Для решения любой нетривиальной задачи, как правило, существует несколько алгоритмов. Из возможных алгоритмов выбирают наилучший по некоторому критерию. Обычно в качестве критерия используют оценку точности решения задачи, либо затраты времени на ее решение, либо некоторый интегральный критерий, включающий оценку точности и затраты времени.



Следование Цикл Ветвление

Рисунок 2- Элементарные блок-схемы

Следование, ветвление и цикл называют базовыми конструкциями структурного программирования. Их особенностью является то, что любая из них имеет только один вход и один выход, поэтому они могут вкладываться друг в друга. Целью использования базовых конструкций является получение программы простой структуры. Такую программу легко читать, отлаживать и, при необходимости, модифицировать. Язык Паскаль способствует созданию хорошо структурированных программ, потому что базовые конструкции в нем реализуются с помощью соответствующих операторов.

Различают алгоритмы линейной, разветвляющейся и циклической структуры.

3. ЛАБОРАТОРНАЯ РАБОТА № 1. ЛИНЕЙНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ

Цель работы: изучение основных типов данных языка Pascal и правил записи констант, арифметических выражений; получение навыков в разработке и отладке линейных программ.

3.1. Теоретические сведения

Алгоритм линейной структуры - алгоритм, в котором блоки выполняются однократно и последовательно друг за другом. Такой порядок выполнения называется естественным. Программа, реализующая такой алгоритм, называется линейной.

Программа на Паскале состоит из необязательного заголовка, разделов описаний и раздела операторов:

```
program имя;   { заголовок }
 раздеы описаний
 begin
  раздел операторов
  end.              (* программа заканчивается точкой *)
```

Программа может содержать комментарии, заключенные в фигурные скобки { } или в скобки вида (* *). Комментарии служат для документирования программы-компилятор их игнорирует, поэтому на их содержимое никаких ограничений не накладывается.

В разделе операторов записываются исполняемые операторы программы. Ключевые слова **begin** и **end** не являются операторами, а служат для их объединения в так называемый составной оператор, или блок. Блок может записываться в любом месте программы, где допустим обычный оператор.

Разделы описаний бывают нескольких видов: описание меток, констант, типов, переменных, процедур и функций.

Любой раздел, кроме раздела операторов, может отсутствовать. Разделителем между разделами и операторами служит точка с запятой. В конце программы должна стоять точка.

Программы линейной структуры обычно используются для вычисления значений выражений.

Выражение — это правило вычисления значения. В выражении участвуют *операнды,* объединенные знаками операций. Операндами выражения могут быть константы, переменные и имена функций. Операции выполняются в определенном порядке в соответствии с приоритетами. Для изменения порядка выполнения операций используются круглые скобки, уровень их вложенности не ограничен. Результатом выражения всегда является значение определенного типа, который зависит от типов операндов. Величины, участвующие в выражении, должны быть совместимых типов. Например, допускается использовать в одном выражении величины целых и вещественных типов. Результат такого выражения будет вещественным.

Ниже приведены операции языка Паскаль, упорядоченные по убыванию приоритетов.

1) Унарная операция not, унарный минус -,взятие адреса@.
2) Операции умножения и деления: *, /, div, mod, and.
3) Операции сложения и вычитания: +, -, or, xor. .
4) Операции отношения: =, <, >, <>, <=, >=.

Функции, используемые в выражении, вычисляются в первую очередь, затем вычисляются выражения в скобках, операции одного приоритета выполняются слева направо в порядке их появления в выражении.

Переменная-это величина, которая в процессе выполнения программы может менять своё значение. Все переменные, используемые в программе, должны быть описаны в разделе переменных, начинающемся со

служебного слова **var.** Для каждой переменной должны быть указаны ее имя и тип.

Константы могут быть именованными и неименованными. Именованные константы должны быть описаны в разделе констант, начинающемся со служебного слова **const.** Константы не могут менять значения в процессе выполнения программы. Константа и переменная являются частными случаями выражения.

Описание некоторых стандартных функций языка Паскаль приведено в приложении А.

Для сохранения вычисленного значения выражения используется **оператор присваивания,** который обозначается знаком **:=** . При выполнении оператора присваивания вычисляется значение выражения, стоящего в правой его части, и это значение присваивается переменной из левой части оператора присваивания.

Рассмотрим пример программы вычисления площади треугольника по трем сторонам A, B, C:

```
Program PRIMER1;
   var A, B, C, S, P: real;
  begin
   read(A,B,C);
   writeln(A,B,C);
   P:=(A+B+C)/2;
   S:=sqrt(P*(P-A)*(P-B)*(P-C));
   writeln('S=',S:8:3)
  end.
```

3.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal, реализующую алгоритм линейной структуры для вычисления значения переменной *f,* в соответствии с вариантом задания. В программе предусмотреть ввод

исходных данных с клавиатуры и вывод результатов на экран.

2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

3.3. Варианты заданий

1) $f = \dfrac{1+\cos^2(x+y)}{\left|x^3 - \dfrac{2y}{(x+y)^2}\right|}$

2) $f = \dfrac{\ln|x|}{\sqrt[3]{|x|+|y|}}$

3) $f = \sqrt{|y|e^{-(y+x)/2}}$

4) $f = \dfrac{x}{\sin arctgz} + \dfrac{z+x}{\cos^2 y}$

5) $f = \sqrt[3]{e^{x-(1/\sin z)}}$

6) $f = x\left(arctgz + e^{-(x+3)}\right)$

7) $f = |x-y|\left(\sin^2 z + tgz\right)$

8) $f = \sqrt[4]{y+\sqrt[3]{x}-1}$

9) $f = x\left(\sin arctgz + \cos^2 y\right)$

10) $f = e^{|x-y|}\left(tg^2 z^2 + 1\right)$

11) $f = \cos^2\left(arctg\, 1/z\right)$

12) $f = 5arctgx - 1/4\, tgy^2$

13) $f = (y-x)\dfrac{y - z/(y-x)}{1+(y-x)^2}$

14) $f = y^x + \sqrt[3]{|x|+|y|}$

15) $f = \lg\left(\sqrt{x}+\sqrt{y}+2\right)$

16) $f = y + \dfrac{x^3}{y + \dfrac{x^2}{y + \dfrac{x^3}{y}}}$

17) $f = \lg\left(\sqrt{e^{x-y}} + x^{|y|} + z\right)$

18) $f = 1 + \dfrac{z^2}{3 + z^2/5} + \sin\dfrac{z^2}{4}$

19) $f = \ln\left(y^{\sqrt{|x|}}\right)\left(x - \dfrac{y}{2}\right)$

20) $f = \dfrac{|\cos x + \cos y|}{\left(1 + 2\sin^2 y^2\right)}$

21) $f = \dfrac{e^x + \cos^2(x+y)}{\left|x - \dfrac{2y}{(x+y)^4}\right|}$

22) $f = \dfrac{\ln|x^3|}{\sqrt[4]{|x|+|y|}}$

23) $f = \sqrt[4]{\sqrt{|y|e^{-(y+x)/2}}}$

24) $f = \dfrac{x}{\sin^2 arctgz} + \dfrac{|z+x|}{\cos^2 y}$

25) $f = \sqrt[3]{e^{x-2}}\cos x^4$

3.4.Содержание отчета

Цель работы, постановка задачи, блок-схема алгоритма, текст программы, тестовые примеры, анализ полученных результатов, выводы.

3.5. Контрольные вопросы

1) Какие типы данных используются в языке программирования Pascal?
2) Указать диапазон значений величин целого и вещественного типов.
3) Какие имена переменных допустимы в программе? Как задать тип переменой в программе?
4) Привести примеры стандартных функций в языке Pascal.
5) Допустимо ли использование величин разных типов в арифметическом выражении? Привести примеры.
6) Указать приоритет выполнения операций при вычислении арифметических выражений.
7) Какова структура программы на языке Pascal?
8) Привести примеры записи констант.

4. ЛАБОРАТОРНАЯ РАБОТА № 2. РАЗВЕТВЛЯЮЩИЕСЯ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ. ИЗУЧЕНИЕ УСЛОВНОГО ОПЕРАТОРА

Цель работы: изучение основных типов данных и условного оператора языка Pascal, получение навыков в разработке и отладке программ разветвляющихся вычислительных процессов.

4.1. Теоретические сведения

На практике линейные программы встречаются крайне редко. Обычно, в зависимости от выполнения каких-либо условий, вычисление осуществляется либо по одним, либо по другим формулам. Алгоритмы такого типа называют алгоритмами разветвляющейся структуры. В общем случае число вариантов продолжения вычислений в алгоритме разветвляющейся структуры произвольно.

Условный оператор - это средство ветвления вычислительного процесса. Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить те или иные действия.

Условный оператор в языке Pascal имеет две разновидности:

IF <выражение> **THEN** <оператор1> **ELSE** <оператор2>;
или
IF < выражение > **THEN** <оператор1> ;

где **IF, THEN,ELSE** - зарезервированные слова,
<выражение>- произвольное выражение логического типа,
<оператор1>,<оператор2>- любые операторы языка Pascal.

Для условного оператора первого вида, если <выражение> принимает значение **TRUE** выполняется <оператор1>, если <выражение> принимает значение

FALSE, то выполняется <оператор2>. Для условного оператора второго вида, если <выражение> принимает значение **TRUE** так же выполняется <оператор1>, но если же <выражение> принимает значение **FALSE,** то выполняется оператор непосредственно следующий за оператором **IF.**

Для формирования сложных условий используются логические операции **not, and, or**.

Рассмотрим пример. Вычислить значение переменной y, если

$$y = \begin{cases} 0, x < -2 \\ -x - 2, -2 \le x < -1 \\ x, -1 \le x < 1 \\ -x + 2, 1 \le x < 2 \\ 0, x \ge 2 \end{cases}.$$

Представим алгоритм в виде последовательности шагов:

1.Ввести значение аргумента x.

2.Вычислить значение y в соответствии с формулами:

если x < -2 или если x ≥ 2, то присвоить переменной y значение 0;

если -2 ≤ x < -1, то присвоить переменной y значение -x - 2;

если -1 ≤ x < 1, то присвоить переменной y значение x;

если 1≤x < 2, то присвоить переменной y значение -x + 2.

3.Вывести значения x и y.

Программа, соответствующая алгоритму, приведена ниже.

```
program PRIMER2;
var x, y : real;
begin
writeln('Введите значение аргумента');
readln(x);
```

```
        if  (x < -2) or(x>=2)       then y := 0;
        if (x >= -2) and (x < -1) then y := -x - 2:
        if (x >= -1) and (x <  1) then y := x;
        if (x >=  1) and (x <  2) then y := -x + 2;
writeln('Если x =',x:6:2,',то значение функции y=',y:6:2)
end.
```

4.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal, реализующую алгоритм разветвляющейся структуры для вычисления значения переменной ***f,*** в соответствии с вариантом задания. В программе предусмотреть ввод исходных данных с клавиатуры и вывод результатов на экран.
2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

4.3. Варианты заданий

1

1) $f = \begin{cases} \cos^2(y+3), x \le z, \\ x + arctgz^2, x > z \end{cases}$

2) $f = \begin{cases} 1 + \frac{x^2}{2} + \frac{y^2}{4}, x \le y, \\ xarctgz^3, x > y \end{cases}$

3) $f = \begin{cases} \sqrt{y + (x+1)^3}, x \ge z, \\ \sin^2 z + tgy, x < z \end{cases}$

4) $f = \begin{cases} e^{x+1} + e^{\frac{x+y}{z}}, x \le y, \\ arctg^2(\ln y) + x, x > y \end{cases}$

5) $f = \begin{cases} \ln^2(x+2), x \le y, \\ \frac{y}{2} + \sin^2 z^2, x > y \end{cases}$

6) $f = \begin{cases} \lg(z+x), x \ge z, \\ tg^2 \frac{y}{2}, x < z \end{cases}$

7) $f = \begin{cases} e^{(x-y)} + z^3, x \ge y, \\ x - \dfrac{x^3}{6}, x < y \end{cases}$

8) $f = \begin{cases} 2\cos(x+z), x \le z, \\ \dfrac{yz^2}{3 + z^3/5}, x > z \end{cases}$

9) $f = \begin{cases} \sqrt{1 + |x-y|^2}, x \le z, \\ tg^2 y^3 + 1, x > z \end{cases}$

10) $f = \begin{cases} 1 + \lg(x+y), x \ge z, \\ \left(arctg \dfrac{z}{2} \right)^3, x < z \end{cases}$

11) $f = \begin{cases} y + 2\sin^2 x, x \ge z, \\ 1 + \dfrac{z}{\sqrt[4]{x+z}}, x < z \end{cases}$

12) $f = \begin{cases} 5arctgz, x \le z, \\ \dfrac{3(x-y)}{z^2 + x^2/z^2}, x > z \end{cases}$

13) $f = \begin{cases} e^{|x-y|}, x > z, \\ \lg z + tgx, x \le z \end{cases}$

14) $f = \begin{cases} \sin^3(y^3 + 1), x \le z, \\ \sqrt{\dfrac{1}{x+y} - y^2}, x > z \end{cases}$

15) $f = \begin{cases} \dfrac{5 + \sqrt{x}}{\sqrt{|y-x|}}, x \ge z, \\ e^{z-1} + \sin^2 \sqrt{x}, x < z \end{cases}$

16) $f = \begin{cases} \sqrt[4]{x+y}, x \ge z, \\ e^{y+x}, x < z \end{cases}$

17) $f = \begin{cases} \dfrac{x+y}{5 + \sin^2 z}, x \ge z, \\ \sqrt{|y + tg^2 z|}, x < z \end{cases}$

18) $f = \begin{cases} \sqrt{|x-y|}, x \ge y, \\ \sin(arctgz), x < y \end{cases}$

19) $f = \begin{cases} e^x + e^z, x \le y, \\ \dfrac{1}{2} arctg(\ln y), x > y \end{cases}$

20) $f = \begin{cases} \dfrac{\cos x}{4 + \sqrt{y}}, x \le y, \\ 2arctg(z+x), x > y \end{cases}$

4.4. Содержание отчета

Цель работы, постановка задачи, структура условного оператора в языке Pascal, блок-схема алгоритма, текст программы, тестовые примеры, анализ полученных результатов, выводы.

4.5. Контрольные вопросы

1) Перечислить действия, реализуемые при выполнении условного оператора.
2) Дать определение вычислительного процесса разветвляющейся структуры.
3) Записать фрагмент программы для вычисления

$$Y = \begin{cases} \ln(x), & \text{если } x >= 1 \\ 1, & \text{если } 1 > x >- 1 \\ 1/x^2, & \text{если } x <= -1. \end{cases}$$

4) Указать приоритеты выполнения логических операций.
5) Какое количество тестовых примеров необходимо для проверки программы разветвляющейся структуры?

5. ЛАБОРАТОРНАЯ РАБОТА № 3. РАЗВЕТВЛЯЮЩИЕСЯ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ. ИЗУЧЕНИЕ ОПЕРАТОРА ВАРИАНТА

Цель работы: изучение оператора варианта языка Pascal, получение навыков в разработке и отладке программ разветвляющихся вычислительных процессов.

5.1. Теоретические сведения

Если необходимо выбирать из более, чем двух возможных вариантов продолжения решения, то используется оператор варианта (выбора) **CASE:**

CASE <выражение-селектор> **OF**
<значение 1>: <оператор1>;
<значение 2>: <оператор2>;
...
<значение n >: <оператор n>;
END,

где <выражение-селектор> - выражение скалярного типа (но не вещественного), определяющее выбор;

<значение i> - конкретное значение или список значений выражения;

<оператор i> - выполняемый оператор.

Если множество значений переменной не исчерпывается приведенным списком значений, после ветви

<значение n >: <оператор n>;

добавляется альтернативное ветвление:

ELSE <оператор>.

Выполнение оператора варианта начинается с вычисления значения <выражения-селектора>. Далее управление передается на оператор, помеченный константой, равной вычисленному значению. Если совпадения не произошло, то выполняется оператор,

стоящий в ветви **ELSE,** а при ее отсутствии - оператор, непосредственно следующий за **CASE.** Заметим, что наличие ветви **ELSE** не обязательно. Однако, рекомендуется всегда обрабатывать ситуацию, когда значение выражения не совпадает ни с одним из указанных значений.

Оператор варианта удобно использовать для ввода и вывода значений перечислимых скалярных типов. Например, в следующем фрагменте с внешнего носителя вводится порядковый номер объекта из списка значений перечислимого типа COLOR. Оператор CASE присваивает соответствующее значение переменной CLR. Аналогично осуществляется вывод значений переменной CLR при помощи оператора варианта.

```
 program PRIMER3;
type color=(red, blue, black);
 var x:integer; clr:color;
    begin
         readln(x);
         case x of
              0:clr:=red;
              1:clr:=blue;
              2:clr:=black;
              else writeln('Введено
  недопустимое значение');
         end;
          . . .
         write('color-');
          case clr of
              red:writeln('red');
               blue:writeln('blue');
               black:writeln('black');
         end
end.
```

5.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal, реализующую алгоритм разветвляющейся структуры в соответствии с вариантом задания. В программе предусмотреть проверку корректности исходных данных и вывод результатов на экран.
2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

5.3.Варианты заданий

1) Для целого числа *k* от 1 до 99 напечатать фразу «мне *k* лет», учитывая при этом, что при некоторых значениях *k* слово «лет» надо заменить на слово «год» или «года».
2) Для натурального числа *k* напечатать фразу «мы нашли *k* грибов в лесу», согласовав окончание слова «гриб» с числом k.
3) Переменной *d* присвоить количество дней в месяце m (год считать невисокосным). Исходные данные задавать в программе в виде констант.
4) Переменной *d* присвоить количество дней в месяце m (год считать невисокосным). Исходные данные вводить в программе с помощью оператора CASE.
5) Переменной t присвоить значение *true,* если тройка y, m, d образует правильную дату, и значение *false* -иначе (при 31 июня и т. п.). Исходные данные задавать в программе в виде констант.
6) Переменной t присвоить значение *true,* если тройка y, m, d образует правильную дату, и значение *false* -иначе (при 31 июня и т. п.). Исходные данные вводить в программе с помощью оператора CASE.
7) По дате (число, месяц, год) определить дату следующего дня. Исходные данные вводить в программе с помощью оператора CASE.

8) По дате (число, месяц, год) определить дату следующего дня. Исходные данные задавать в программе в виде констант.
9) По заданной дате (число, месяц) определить порядковый номер дня високосного года. Исходные данные вводить в программе с помощью оператора CASE.
10) По заданной дате (число, месяц) определить порядковый номер дня високосного года. Исходные данные задавать в программе в виде констант.
11) По номеру дня високосного года определить его дату (число, месяц).
12) По заданному символу определить его тип: буква, цифра, специальный символ.
13) По заданному символу определить его принадлежность к латинскому алфавиту или к латинскому и русскому алфавитам.
14) Считая, что год невисокосный и 1 января этого года приходится на известный день недели, определить день недели дня по заданной дате (число, месяц). Исходные данные задавать в программе в виде констант.
15) Считая, что год невисокосный и 1 января этого года приходится на известный день недели, определить день недели дня по заданной дате (число, месяц). Исходные данные вводить в программе с помощью оператора CASE.
16) Значение переменной X, означающее длину в единицах P (дециметр, километр, миллиметр, сантиметр, метр), заменить на длину этой величины в метрах. Исходные данные задавать в программе в виде констант.
17) Значение переменной X, означающее длину в единицах P (дециметр, километр, миллиметр, сантиметр, метр), заменить на длину этой величины в метрах. Исходные данные вводить в программе с помощью оператора CASE.
18) Напечатать значение переменной k римскими цифрами.
19) Значение переменной X, означающее вес в единицах P (грамм, килограмм, пуд, тонна,), заменить на вес этой величины

в килограммах. Исходные данные задавать в программе в виде констант.

20) Значение переменной X, означающее вес в единицах P (грамм, килограмм, пуд, тонна,), заменить на вес этой величины в килограммах. Исходные данные вводить в программе с помощью оператора CASE.

5.4. Содержание отчета

Цель работы, постановка задачи, структура оператора варианта в языке Pascal, блок-схема алгоритма, текст программы, тестовые примеры, анализ полученных результатов, выводы.

5.5. Контрольные вопросы

1) Указать структуру оператора варианта.
2) Какой тип может иметь переменная-селектор?
3) На какое количество ветвей можно выполнить разветвление с помощью оператора варианта?
4) Как выбирать значения исходных данных для тестирования?
5) Какое количество тестовых примеров необходимо для проверки программы?

6. ЛАБОРАТОРНАЯ РАБОТА № 4. ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ. ВЫЧИСЛЕНИЕ СУММЫ РЯДА

Цель работы: изучение операторов цикла языка Pascal, получение навыков в разработке и отладке программ циклической структуры.

6.1. Теоретические сведения

Часто при решении задач на практике приходится многократно выполнять одни и те же действия для разных значений некоторых переменных. Такие многократно повторяемые участки вычислительного процесса называются циклами. Использование циклов позволяет существенно сократить объем программы. Для организации цикла необходимо задать начальное значение переменной, изменяющейся в цикле (параметр цикла); изменять переменную перед каждым новым повторением цикла; проверять условие окончания или повторения цикла.

Различают циклы с заданным числом повторений и итерационные циклы,. В языке Pascal имеются три типа операторов цикла: оператор цикла **FOR** с заданным числом повторений, оператор цикла **WHILE** с предусловием, оператор цикла **REPEAT… UNTIL** с постусловием. Последние два относятся к итерационным циклам.

Оператор цикла **WHILE** с предусловием имеет вид:

WHILE <выражение> **DO** <оператор>.

Здесь < выражение > - выражение логического типа,

<оператор> - произвольный оператор языка Pascal, составляющий тело цикла.

Выполнение оператора начинается с вычисления значения выражения. Если оно имеет значение **TRUE,** то

выполняется тело цикла. Затем выполнение оператора цикла повторяется до тех пор, пока значение выражения не станет равным **FALSE**. Как только получается значение **FALSE**, управление передается оператору, следующему за оператором цикла, а оператор внутри цикла не выполняется. Если выражение булевского типа было ложно при первом входе в цикл, то тело цикла не выполняется ни разу. Очевидно, что один из операторов, находящихся внутри цикла, должен влиять на значение выражения, поскольку иначе цикл будет повторяться бесконечно.

Оператор цикла с постусловием похож на оператор цикла с предусловием, но условие вычисляется и проверяется после выполнения операторов, составляющих тело цикла. Общий вид оператора цикла с постусловием:

REPEAT <тело_цикла> **UNTIL** < выражение >.
Здесь **REPEAT, UNTIL** - зарезервированные слова,
<тело_цикла>- произвольная последовательность операторов Pascal,
< выражение > - выражение логического типа.

Оператор цикла с постусловием начинается с выполнения операторов внутри цикла. Затем вычисляется выражение, и если получается истинное значение, то осуществляется выход из цикла. Если же значение выражения ложно, то выполнение операторов, составляющих тело цикла, повторяется, а затем снова вычисляется выражение.

Отметим, что в отличие от цикла с предусловием выход из цикла с постусловием осуществляется при истинности выражения. Очевидно, что цикл с постусловием выполняется хотя бы один раз.

Для всех операторов цикла выход из цикла может осуществляться как вследствие естественного

окончания цикла, так и с помощью операторов перехода и выхода. В версии ТУРБО ПАСКАЛЬ 7.0 определены стандартные процедуры **Break** и **Continue**. Процедура **Break** выполняет безусловный выход из цикла. Процедура **Continue** обеспечивает переход к началу новой итерации цикла.

Рассмотрим пример.Найти сумму ряда с точностью до $\xi=10^{-5}$, если общий член ряда $a_n = 2\times(n!)^2/(3\times(2\times n)!)$.

Для получения рекуррентной формулы найдем отношение a_{n+1}/a_n:

$a_{n+1}/a_n = (n+1)/(2\times(2\times n+1))$.

Тогда $a_{n+1} = a_n \times (n+1)/(2\times(2\times n+1))$.

При составлении программы будем считать, что заданная точность достигнута, если $a_n < \xi$.

```
program PRIMER4;
const eps=0.1e-4;
var n:integer; An,Summa:real;
begin
        Summa:=0; n:=1; An:=1/3;
        while An>eps do
        begin
          Summa:=Summa+An;
          n:=n+1;
          An:=An*(n+1)/2/(2*n+1)
          end;
writeln('Сумма=', Summa,' количество слагаемых=',n)
end.
```

6.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal, реализующую алгоритм вычисления суммы бесконечного ряда с точностью до $\xi=10^{-4}$, в соответствии с вариантом задания. В программе предусмотреть:

- ввод исходных данных с клавиатуры,
- проверку их корректности и вывод результатов на экран;
- вычисление точного значения заданной функции и количества слагаемых для достижения указанной точности.

Программу представить в двух вариантах: в первом варианте использовать оператор цикла WHILE, во втором-REPEAT.

2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

6.3. Варианты заданий

1) $$\ln(\frac{x+1}{x-1}) = 2(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \frac{1}{7x^7} + ...)$$

$(|x| > 1)$

2) $$\ln(\frac{1+x}{1-x}) = 2(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + ...)$$

$(|x| < 1)$

3) $$shx = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + ...$$

4) $$\ln x = 2\left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \ldots\right)$$

$(x > 0)$

5) $$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$$

$(-1 < x \le 1)$

$$arctgx = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

6)

$(|x| < 1)$

7) $$\ln(x) = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \ldots$$

$(0 < x \le 2)$

8) $$\ln(x) = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \ldots$$

$(x > \frac{1}{2})$

9) $$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

10) $$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

11) $$e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$$

12) $$\frac{\sin x}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \ldots$$

13) $e^{-x^2} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \ldots$

14) $arctgx = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \ldots$

x>1

15) $\ln x = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \ldots$

x>1/2

16) $\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \ldots$

-1≤x<1

6.4. Содержание отчета

Цель работы, постановка задачи, структура операторов цикла в языке Pascal, блок-схема алгоритма, описание используемых переменных, текст программы, тестовые примеры, анализ полученных результатов, выводы.

6.5. Контрольные вопросы

1) Указать последовательность действий, выполняемых при организации циклических участков программы.
2) Указать назначение и правила организации цикла.
3) Указать отличия в организации циклов с заданным числом повторений и итерационных циклов.
4) Указать отличия циклов типа WHILE от циклов типа REPEAT.
5) В чем состоят преимущества использования операторов цикла в программах?
6) Указать назначение стандартных процедур Break и Continue.

7. ЛАБОРАТОРНАЯ РАБОТА № 5. ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ. ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ

Цель работы: изучение операторов цикла языка Pascal, получение навыков обработки массивов данных и отладки программ циклической структуры.

7.1.Теоретические сведения

Совокупность данных одного типа, расположенных в смежных ячейках памяти, называют массивом. Обращение к элементу массива осуществляется по имени и по индексу. Индекс- это номер элемента в массиве. Массив описывается при помощи задания типа его элементов, указания количества и типа индексов.

Общий вид описания массива **ARRAY[t1] OF t2**, где **ARRAY**-зарезервированное слово, **t1**-тип индекса, **t2**- тип элементов массива. Тип индекса- любой скалярный или ограниченный тип. Тип элементов массивов может быть любым. Так, если **t2** опять задает массив, то про исходный массив говорят, что он многомерный. Вещественный тип нельзя использовать для задания типа индексов элементов массива. На практике обработка массивов данных, как правило, сводится к выполнению одних и тех же действий с каждым элементом массива. Поэтому при разработке соответствующих программ наиболее часто используют циклы с заданным числом повторений.

Оператор цикла **FOR** имеет структуру:

FOR <пар_цик> := <нач_знач> **TO** <кон_знач> **DO** <оператор>

или

FOR <пар_цик> := <нач_знач> **DOWNTO** <кон_знач> **DO** <оператор>

Здесь **FOR, TO, DO, DOWNTO** - зарезервированные слова,

<пар_цик> - параметр цикла,

<нач_цик> - начальное значение параметра цикла,

<кон_цик> - конечное значение параметра цикла,

<оператор> - произвольный оператор языка Pascal.

В первом случае значение параметра цикла увеличивается на 1 при каждом проходе цикла, во втором – на 1 уменьшается.

Тип параметра цикла обязательно должен совпадать с типом начального и конечного значений цикла. Из стандартных скалярных типов в качестве параметра цикла нельзя использовать переменные вещественного типа.

Выполнение оператора цикла начинается с проверки условия **<пар_цик> < =<кон_знач>** для цикла **TO** (или **<пар_цик> > =<кон_знач>** для цикла **DOWNTO**). Если оно не справедливо, то оператор, составляющий тело цикла, не выполняется, а управление передается следующему за циклом оператору. Если же условие истинно, то выполняется тело цикла, а затем параметру цикла присваивается следующее значение P:=SUCC(P) (для цикла TO) или предыдущее значение P:=PRED(P) (для цикла DOWNTO). Далее весь процесс повторяется. Если параметр цикла целого типа, то это означает его увеличение (соответственно уменьшение) на единицу при каждом новом выполнении расположенного в цикле оператора. Задать шаг, отличный от 1 или - 1, нельзя!

Вводить и выводить массивы можно только поэлементно. Над массивами в целом допустима только операция присваивания, при этом массивы должны быть одинакового типа.

Размещение массива в памяти происходит до выполнения программы, поэтому при описании индексов можно применять только константы или константные выражения. Использовать для этого переменные нельзя.

Рассмотрим пример сортировки (упорядочивания) элементов массива из 10 целых чисел по возрастанию. Существует множество методов сортировки [6].

Мы для сортировки будем использовать *метод прямого выбора.* Его сущность заключается в следующем. Выбирается наименьший элемент массива и меняется местами с первым элементом, затем просматриваются элементы, начиная со второго, и наименьший из них меняется местами со вторым элементом, и так далее n - 1 раз. На последнем, n-ом проходе цикла при необходимости меняются местами предпоследний и последний элементы массива.

```
program PRIMER5;        { Сортировка выбором }
  const n = 10;
  var a : array [1 .. n] of integer;
  I,j, nmin, min ; integer;
    begin
    writeln('Введите ', n, ' элементов массива');
      for i := 1 to n do
        read(a[i]);
      { просмотр массива n - 1 раз }
      for i := 1 to n - 1 do
          begin  nmin := i; min:=a[i];
      { поиск наименьшего элемента }
            for j := i + 1 to n do
             if a[j] < min then
              begin nmin := j; min:=a[nmin] end;
            a[nmin]:=a[i];
            a[i] := min;
        end;
        writeln('Упорядоченный  массив: ');
        for i := 1 to n do
         write (a[i], ' ')
        end.
```

7.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal, в которой ввести одномерный массив X,

содержащий N элементов заданного типа, выполнить обработку в соответствии с вариантом. Исходный массив и результаты вывести на экран.
2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

7.3. Варианты заданий

1) Найти сумму отрицательных и произведение положительных элементов массива.
2) Переставить минимальный и максимальный элементы массива.
3) Упорядочить элементы массива по возрастанию.
4) Упорядочить элементы массива по убыванию.
5) Найти среднее арифметическое всех положительных элементов массива.
6) Определить количество положительных и количество отрицательных элементов массива, абсолютная величина которых не превосходит X.
7) Преобразовать массив, расположив в нем элементы в обратной последовательности.
8) Преобразовать массив, заменив все отрицательные числа на -1,а положительные на +1.
9) Поменять местами первый и максимальный элементы массива.
10) Найти минимальный и максимальный элементы массива и их порядковые номера.
11) Найти среднее арифметическое всех отрицательных элементов массива, расположенных после первого нулевого элемента.
12) Определить количество положительных, количество отрицательных и количество нулевых элементов массива.
13) “Сжать” массив, удалив из него все нулевые элементы.

14) Преобразовать массив, поменяв первый элемент с последним, второй с предпоследним и т.д.
15) Преобразовать массив, расположив сначала все положительные, а затем все отрицательные элементы. Порядок следования элементов одного типа не менять.
16) Найти произведение всех положительных элементов массива, расположенных после первого нулевого элемента.
17) Поменять местами минимальный элемент массива и первый положительный элемент.
18) Определить количество элементов массива кратных 5, значения которых не превосходят X.
19) Найти среднее арифметическое элементов массива, лежащих в диапазоне от A до B.
20) Определить максимальный по абсолютной величине элемент массива и переставить его с последним нулевым элементом массива.

7.4. Содержание отчета

Цель работы, постановка задачи, структура операторов цикла в языке Pascal, блок-схема алгоритма, текст программы, тестовые примеры, анализ полученных результатов, выводы.

7.5. Контрольные вопросы

1) Дать определение регулярных структур данных.
2) Указать способы описания одномерных и многомерных массивов. Привести примеры.
3) Указать правила организации циклов с заданным числом повторений.
4) Как осуществляется доступ к элементам массива?
5) Какой тип может иметь переменная цикла при использовании циклов с заданным числом повторений?

8. ЛАБОРАТОРНАЯ РАБОТА № 6. ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ. ОБРАБОТКА ДВУМЕРНЫХ МАССИВОВ

Цель работы: изучение операторов цикла языка Pascal, получение навыков обработки двумерных массивов данных и отладки программ циклической структуры.

8.1. Теоретические сведения

Общий вид описания массива **ARRAY[t1] OF t2,** где **ARRAY,OF**-зарезервированные слова, **t1**-тип индекса, **t2**-тип элементов массива. Так, если **t2** опять задает массив, то про исходный массив говорят, что он многомерный.

Размерность многомерного массива (количество индексов) может быть любой.

Для описания многомерных массивов часто используют сокращенную форму

ARRAY[t1,t2,…tn] OF tm, где **ti**- тип **i-**го индекса, **tm**-тип элементов массива.

Если размерность массива равна 2, то имеем дело с матрицей. В памяти ЭВМ элементы матрицы хранятся по строкам: сначала элементы первой строки, затем второй и т.д. Однако, в памяти компьютера строки ничем не отделены одна от другой. Обращение к элементам двумерного массива осуществляется по имени и по индексам: номеру строки и столбца, на пересечении которых он находится, причем первым указывается номер строки, а вторым- номер столбца.

Рассмотрим пример. Пусть задан двумерный массив различных вещественных чисел, содержащий 5 строк и 4столбца. Строку, содержащую максимальный элемент массива, поменять местами со строкой, содержащей минимальный элемент.

Введем обозначения:

m- исходный массив, содержащий 5 строк и 4столбца;
max- максимальный элемент массива
min -минимальный элемент массива
maxi- номер строки, в которой находится максимальный элемент массива;
mini- номер строки, в которой находится минимальный элемент массива;
m1- вспомогательный одномерный массив, выполняющий роль буфера при перестановке строк.

```
program PRIMER6;
type mas=array[1..4] of real;
var m: array[1. .5] of мas; m1:мas;
max, min: real; maxi, mini, i, j: integer;
    begin
    { Ввод элементов матрицы}
        writeln('введите исходный массив');
        for i:=1 to 5 do
            for j:=1 to 4 do read(m[i,j]);
   {Поиск максимального и минимального
элементов массива}
        max:=m[1,1]; min:=m[1,1]; maxi:=1; mini:=1;
        for i:=1 to 5 do
            for j:=l to 4 do
                begin
                        if max<m[i,j] then begin
max:=m[i,j]; maxi:=i end;
                        if min>m[i,j] then begin
min:=m[i,j]; mini:=i end
                end;
   { Перестановка элементов строк с номерами
maxi  и mini}
        m1:=m[maxi];
        m[maxi]:=m[mini];
        m[mini]:=m1;
        writeln('Массив после перестановки строк');
```

```
    { Печать двумерного массива в виде таблицы
}
        for i:=1 to 5 do
          begin
            for j:=1 to 4 do
            write(m[i,j]:5:2);
            writeln
          end
end.
```

8.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal, в которой ввести двумерный массив X, содержащий N строк и M столбцов элементов заданного типа; выполнить обработку по варианту. Исходный массив и результаты вывести на экран, при этом двумерный массив выводить в форме матрицы.
2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

8.3. Варианты заданий

1) Преобразовать массив: элементы строки, в которой находится максимальный элемент матрицы, заменить нулями.
2) Преобразовать массив: элементы того столбца, в котором находится максимальный элемент матрицы, заменить на нули.
3) Каждый столбец массива упорядочить по убыванию.
4) Сформировать одномерный массив, состоящий из максимальных элементов каждого столбца.

5) Преобразовать массив: разделить элементы каждого столбца заданной матрицы на последний элемент столбца.
6) Преобразовать массив: разделить элементы каждой строки матрицы на последний элемент этой строки.
7) Определить количество нулевых элементов в каждой строке матрицы.
8) Определить количество нулевых элементов в каждом столбце матрицы.
9) Сформировать одномерный массив, состоящий из сумм положительных элементов каждого столбца.
10) Сформировать одномерный массив, состоящий из сумм отрицательных элементов каждой строки.
11) Найти среднее арифметическое положительных элементов каждой строки.
12) Найти среднее арифметическое положительных элементов каждого столбца.
13) Преобразовать массив, умножив элементы каждой строки на минимальный элемент этой строки.
14) Преобразовать массив, умножив элементы каждого столбца на минимальный элемент этого столбца.
15) Найти количество нулевых элементов в каждой строке матрицы.
16) Найти количество нулевых элементов в каждом столбце матрицы.
17) Преобразовать массив: разделить элементы каждой строки матрицы на абсолютную величину первого элемента этой строки.
18) Преобразовать массив: разделить элементы каждой строки матрицы на среднее арифметическое элементов этой строки.

19) Преобразовать массив: умножить элементы каждого столбца матрицы на среднее арифметическое элементов этого столбца.
20) Сформировать одномерный массив, состоящий из средних арифметических положительных элементов каждой строки.

8.4. Содержание отчета

Цель работы, постановка задачи, блок-схема алгоритма, описание переменных, текст программы, тестовые примеры, анализ полученных результатов, выводы.

8.5. Контрольные вопросы

1) Указать способы описания одномерных и многомерных массивов.
2) Как осуществляется доступ к элементам многомерных массивов?
3) Как хранятся в памяти ЭВМ элементы многомерных массивов?
4) Какой тип может иметь переменная цикла при использовании циклов с заданным числом повторений?
5) Какой тип может иметь индекс элемента массива?

9. ЛАБОРАТОРНАЯРАБОТА№7.ОРГАНИЗАЦИЯ ПРОГРАММ, СОДЕРЖАЩИХ ПОДПРОГРАММЫ

Цель работы: изучение принципов разработки программ, содержащих подпрограммы, получение практических навыков разработки и программирования вычислительного процесса сложной структуры, получение навыков по отладке и тестированию подпрограмм.

9.1. Теоретические сведения

При решении сложной задачи рекомендуется разбивать ее на отдельные части, программировать их, а затем объединять в единую программу. Группа операторов, оформленная специальным образом, называется подпрограммой. Использование подпрограмм упрощает процесс программирования и отладки программ. Любая программа может содержать несколько подпрограмм. Каждая подпрограмма может вызываться из любой части программы. Вызов подпрограммы осуществляется по имени. После ее выполнения управление передается в ту часть программы, откуда был сделан вызов. Раздел описания подпрограмм расположен после раздела описания переменных

В языке Паскаль выделяют два вида подпрограмм: подпрограммы-процедуры и подпрограммы-функции.

Общая форма записи процедуры имеет вид:

```
procedure  ИМЯ (<список формальных параметров>);
        < раздел описаний >
      begin
       <раздел операторов>
      end;
```

Общая форма записи подпрограммы-функции имеет вид :

```
function  ИМЯ (<список формальных
параметров>):тип результата ;
        < раздел описаний >
        begin
         <раздел операторов>
        ИМЯ := значение;
        end;
```

Формальные параметры - это параметры, которые определяются в заголовке подпрограммы; а при каждом вызове подпрограммы они заменяются фактическими параметрами. Формальные и фактические параметры должны соответствовать по количеству, типу и порядку следования.

Список формальных параметров может содержать параметры- значения, параметры – переменные, параметры – процедуры и параметры – функции. В последних двух случаях параметры являются именами процедур и функций. Список формальных параметров может отсутствовать.

Рассмотрим параметры первых двух видов.

Описание формальных параметров имеет вид:

- параметр1, параметр2, ... : тип - для параметров-значений;
- var параметр1, параметр2, ... : тип - для параметров - переменных.

Подпрограмма-функция используется в тех случаях, когда результатом выполнения подпрограммы является одно значение. Таким образом, подпрограмма-функция возвращает одно значение, подпрограмма-процедура может не возвращать или возвращать любое количество значений.

Параметры-значения передаются в подпрограмму по значению, а параметры-переменные – по адресу. Если некоторый параметр предназначен для возврата значения

из подпрограммы, то такой параметр обязательно должен быть параметром-переменной.

Напишем функцию **STEP** для возведения вещественного числа в целую степень. Введем обозначения:
a- вещественное число, основание степени;
b- целое число, показатель степени;
rez- переменная для получения значения $a^{|b|}$.

```
function STEP(a:real; в:integer):real;
var rez:real; absb:integer;
begin
  rez:=1; absb:=abs(b);
  while absb>0 do
       begin rez:=rez*a;
             absb:=absb-1
       end;
  if b<0 then rez:=1/rez;
  step:=rez
end;
```

Для решения этой задачи можно использовать и процедуру. В этом случае для передачи результата необходим еще один параметр **rez**. Сохраним введенные обозначения. Текст процедуры приведен ниже.

```
procedure STEP(a:real; в:integer; var rez:real);
var absb:integer;
begin
  rez:=1; absb:=abs(b);
  while absb>0 do
       begin rez:=rez*a;
             absb:=absb-1
       end;
  if b<0 then rez:=1/rez
end;
```

9.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal, в которой ввести массив X, содержащий элементы заданного типа; выполнить обработку по варианту. Исходный массив и результаты вывести на экран.
2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

9.3. Варианты заданий

Вариант 1
В одномерном массиве, состоящем из n вещественных элементов, вычислить:
1) сумму отрицательных элементов массива(оформить в виде функции);
2) произведение элементов массива, расположенных между максимальным и минимальным элементами (оформить в виде процедуры).
Упорядочить элементы массива по возрастанию.

Вариант 2
В одномерном массиве, состоящем из n вещественных элементов, вычислить:
1) сумму нечетных положительных элементов массива(оформить в виде функции);
2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами(оформить в виде процедуры).
Упорядочить элементы массива по убыванию.

Вариант 3
В одномерном массиве, состоящем из п целых элементов, вычислить:
1) произведение элементов массива с четными номерами(оформить в виде функции);
2) сумму элементов массива, расположенных между первым и последним нулевыми элементами(оформить в виде процедуры).
Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом — все отрицательные (элементы, равные 0, считать! положительными).

Вариант 4
В одномерном массиве, состоящем из п вещественных элементов, вычислить;
1) сумму элементов массива с нечетными номерами(оформить в виде функции);
2) сумму элементов массива, расположенных между первым и последним отрицательными элементами(оформить в виде процедуры).
Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 5
В одномерном массиве, состоящем из п вещественных элементов, вычислить:
1) максимальный элемент массива(оформить в виде функции);
2) сумму элементов массива, расположенных до последнего положительного элемента(оформить в виде процедуры).
Сжать массив, удалив из него все элементы, модуль которых находится в интервале [а,Ь]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 6

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) минимальный элемент массива(оформить в виде функции);
2) сумму элементов массива, расположенных между первым и последним положительными элементами(оформить в виде процедуры).

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом — все остальные.

Вариант 7

В одномерном массиве, состоящем из п целых элементов, вычислить:

1) номер максимального элемента массива(оформить в виде функции);
2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами(оформить в виде процедуры).

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине — элементы, стоявшие в четных позициях.

Вариант 8

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) номер минимального элемента массива(оформить в виде функции);
2) сумму элементов массива, расположенных между первым и вторым отрицательными элементами(оформить в виде процедуры).

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом — все остальные.

Вариант 9
В одномерном массиве, состоящем из п вещественных элементов, вычислить:
1) максимальный по модулю элемент массива(оформить в виде функции);
2) сумму элементов массива, расположенных между первым и вторым положительными элементами(оформить в виде процедуры).
Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

Вариант 10
В одномерном массиве, состоящем из п целых элементов, вычислить:
1) минимальный по модулю элемент массива(оформить в виде функции);
2) сумму модулей элементов массива, расположенных после первого элемента, равного нулю(оформить в виде процедуры).
Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине — элементы, стоявшие в нечетных позициях.

Вариант 11
В одномерном массиве, состоящем из п вещественных элементов, вычислить;
1) номер минимального по модулю элемента массива(оформить в виде функции);
2) сумму модулей элементов массива, расположенных после первого отрицательного элемента(оформить в виде процедуры).
Сжать массив, удалив из него все элементы, величина которых находится в интервале [а,Ь]. Освободившиеся в конце массива элементы заполнить нулями:

Вариант 12
В одномерном массиве, состоящем из п вещественных элементов, вычислить:
1) номер максимального по модулю элемента массива(оформить в виде функции);
2) сумму элементов массива, расположенных после первого положительного элемента(оформить в виде процедуры).
Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [a,Ь], а потом — все остальные.

Вариант 13
В одномерном массиве, состоящем из п вещественных элементов, вычислить:
1) произведение положительных элементов массива(оформить в виде функции);
2) сумму элементов массива, расположенных между максимальным и минимальным элементами (оформить в виде процедуры).
Упорядочить элементы массива по убыванию.

Вариант 14
В одномерном массиве, состоящем из п вещественных элементов, вычислить:
1) произведение нечетных положительных элементов массива(оформить в виде функции);
2) сумму элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами(оформить в виде процедуры).
Упорядочить элементы массива по возрастанию.

Вариант 15
В одномерном массиве, состоящем из п целых элементов, вычислить:

1) сумму элементов массива с четными номерами(оформить в виде функции);
2) произведение элементов массива, расположенных между первым и последним нулевыми элементами(оформить в виде процедуры).
Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 16
В одномерном массиве, состоящем из п вещественных элементов, вычислить:
1) сумму элементов массива, превышающих по абсолютной величине значение X (оформить в виде функции);
2) сумму элементов массива, расположенных между первым и последним отрицательными элементами(оформить в виде процедуры).
Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом — все отрицательные (элементы, равные 0, считать! положительными).

Вариант 17
В одномерном массиве, состоящем из п вещественных элементов, вычислить:
1) максимальный по абсолютной величине элемент массива(оформить в виде функции);
2) сумму элементов массива, расположенных до последнего положительного элемента (оформить в виде процедуры).
Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом — все остальные.

Вариант 18

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) минимальный элемент массива (оформить в виде функции);

2) сумму элементов массива, расположенных между первым и последним отрицательными элементами(оформить в виде процедуры).

Сжать массив, удалив из него все элементы, модуль которых находится в интервале [а,Ь]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 19

В одномерном массиве, состоящем из п целых элементов, вычислить:

1) номер минимального элемента массива(оформить в виде функции);

2) произведение элементов массива, расположенных между первым и вторым ненулевыми элементами (оформить в виде процедуры).

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом — все остальные.

Вариант 20

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) номер минимального элемента массива(оформить в виде функции);

2) сумму элементов массива, расположенных между первым и вторым отрицательными элементами(оформить в виде процедуры).

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине — элементы, стоявшие в четных позициях.

9.4. Содержание отчета

Цель работы, постановка задачи, общие формы записи процедур и функций в языке Pascal, блок-схема алгоритма, спецификация процедур и функций, текст программы, тестовые примеры, анализ полученных результатов, выводы.

9.5. Контрольные вопросы

1) Что такое подпрограмма?
2) Указать способы передачи параметров в подпрограмму. Привести примеры.
3) Указать особенности оформления подпрограмм в виде процедуры. Привести примеры.
4) Указать особенности оформления подпрограмм в виде функции. Привести примеры.
5) Какая существует связь между формальными и фактическими параметрами?
6) Каким образом осуществляется вызов процедуры?
7) В чем особенность вызова функции?

10. ЛАБОРАТОРНАЯ РАБОТА №8. ОБРАБОТКА ТЕКСТОВЫХ ФАЙЛОВ

Цель работы: изучение принципов организации файлов, правил работы с текстовыми файлами, а также основных процедур и функций для обработки текстовых файлов.

10.1. Теоретические сведения

Файл – именованный набор данных, состоящий из последовательности компонент одного типа. Число компонент называется длиной файла.

Общий вид описания файлого типа:

type <имя> = file of <тип_компонент>;

где **<имя>** - имя типа,

<тип_компонент>- тип компонентов файла, может быть любым, но не файловым.

Число компонентов в определении файлового типа не фиксируется.

Определим файловый тип FINT следующим образом:

type FINT **= file of** integer;

тогда любая переменная типа FINT – это файл компонентов типа **integer**.

Файлы, компонентами которых являются символы, называются текстовыми. Текстовые файлы представляют собой последовательность строк символов переменной длины. Тип **Text** называется текстовым, является стандартным и определен как **type Text= file of char**.

Объявление текстовых файлов в программе осуществляется вводом переменной типа **text**.

Например:

```
var     file1 : text;
var     file2 : file of char;
```

При занесении информации в файл признак конца строки автоматически записывается в файл с помощью процедуры **writeln**.

Для обнаружения признака конца строки при чтении используется функция **eoln**(**имя файловой переменной**), принимающая значение **TRUE,** если из указанного файла считан символ, за которым следует признак конца строки, и значение **FALSE** в противном случае.

Функция **eof**(**имя файловой переменной)** возвращает значение **TRUE**, если указатель файла установлен на его конец, и **FALSE** в противном случае.

При выполнении операций с файлом выделяют несколько этапов:

- описание файловой переменной;
- открытие файла;
- работа с файлом;
- закрытие файла.

В программе каждой файловой переменной следует назначить имя внешнего файла. Для этого используется процедура

Assign (var имя_файла; внешнее_имя:string).

Для открытия файлов в языке Pascal имеются две процедуры:

Reset (имя файловой переменной); {открыть файл для чтения};

Rewrite (имя файловой переменной); {открыть файл для записи}.

Процедрура **Reset (имя файловой переменной)** устанавливает указатель файла в начальное положение и подготавливает файл для чтения. Действие стандартной процедуры **Rewrite (имя файловой**

переменной) заключается в следующем: текущее значение файла стирается и можно формировать файл.

Процедура **Read (имя файловой переменной, список аргументов)** выполняет чтение данных из файла, а процедура **Write (имя файловой переменной, список аргументов)** - запись данных в файл. (Для текстовых файлов могут использоваться процедуры Readln и Writeln, которые соответственно производят чтение, запись с новой строки).

Процедура **Close** (**Имя файловой переменной**) закрывает файл.

Язык Turbo Pascal располагает рядом функций и процедур для обработки строк. Длина строки не может превышать 255 символов.

1.Функция ***length*** возвращает длину строки **S**:

function Length(S: String): Integer;

2. Функция ***pos*** ищет подстроку **Substr** в строке **S:**

function Pos(Substr: String; S: String): Byte;

3. Функция ***copy*** возращает фрагмент строки **S,** начиная с символа **Index,** длиной **Count**:

function Copy(S: String; Index: Integer; Count: Integer): String;

4. Процедура ***Delete*** удаляет подстроку начиная с позиции **Index** длиной **Count** из строки **S**:

procedure Delete(var S: String; Index: Integer; Count:Integer);

5. Процедура ***Insert*** вставляет в строку **S** подстроку **Source** начиная с позиции **Index:**

procedure Insert(Source: String; var S: String; Index: Integer);

Ниже приведен пример программы, вычисляющий сумму длин всех строк в файле data.txt.

```
Program PRIMER8;
 var   f : text;
   l : longint;
```

```
  str : string;
begin
  l:=0;
  assign(f,'data.txt');
  reset(f);
  while not eof(f) do
  begin
    readln(f,str);
    l:=l+length(str);
  end;
  close(f);
  writeln('Общая длина файла  ',l, ' символов.')
end.
```

10.2. Порядок выполнения работы

1) Разработать блок-схему и программу на языке Pascal для обработки, в соответствии с вариантом, текста из внешнего текстового файла. Результаты обработки поместить также во внешний файл. Для обработки текста использовать строковые функции. (В качестве исходного текста можно использовать программу к лабораторной работе 7.)
2) Разработать тестовые примеры.
3) Отладить программу.
4) Проанализировать полученные результаты.
5) Оформить отчет.

10.3. Варианты заданий

1) Определить количество слов в каждой строке. Сжать каждую строку, удалив лишние пробелы
2) Определить количество строк в исходном тексте. Удалить в каждой строке все четырехбуквенные слова.
3) Определить количество слов в каждой строке, сжать каждую строку, удалив все пробелы.

4) Определить количество строк в исходном тексте. Заменить в каждой строке все первые слова на AAAAA.
5) Определить количество слов в каждой строке, Вывести самое короткое слово каждой строки.
6) Определить количество строк в исходном тексте. Заменить в каждой строке все четырехбуквенные слова на !!!!.
7) Определить количество слов в каждой строке, Сжать каждую строку, удалив из нее все слова BEGIN.
8) Определить количество слов в каждой строке. сжать каждую строку, удалив END в каждой строке.
9) Определить количество строк в исходном тексте. Удалить в каждой строке все трехбуквенные слова.
10) Определить количество слов в каждой строке, сжать каждую строку, удалив все символы **;**.
11) Определить количество строк в исходном тексте. Удалить в каждой строке все последние слова.
12) Определить количество слов в каждой строке, Вывести все трехсимвольные слова каждой строки.
13) Определить количество строк в исходном тексте. Проверить имеется ли в каждой строке баланс скобок.
14) Определить количество строк в исходном тексте. Удалить в каждой строке все четырехбуквенные слова.
15) Определить количество слов в каждой строке. Удалить все слова BEGIN в каждой строке.
16) Определить количество строк в исходном тексте. Удалить в каждой строке все вторые слова.
17) Определить количество строк в исходном тексте. Удалить в каждой строке все предпоследние слова.
18) Определить количество строк в исходном тексте. Удалить в каждой строке все четырехбуквенные слова.
19) Определить количество слов в каждой строке. Вывести самое длинное слово каждой строки.

20) Определить количество слов в каждой строке. Определить количество символов “ ; “ в исходном тексте.

10.4. Содержание отчета

Цель работы, постановка задачи, основные процедуры и функции для обработки текстового файла, блок-схема алгоритма, текст программы, тестовые примеры, анализ полученных результатов, выводы.

10.5. Контрольные вопросы

1) Дать определение файла.
2) Чем отличается текстовый файл от других файлов?
3) Как распознать конец файла?
4) Каким образом выполняется запись данных в файл и чтение данных из файла?
5) Привести примеры описания файлов различных типов.
6) Какие функции используются для работы со строковыми данными?
7) Привести примеры применения функций для обработки строк.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Брукшир Дж. Гленн. Введение в компьютерные науки/ Дж. Гленн Брукшир.- М.: Издательский дом «Вильямс»,2001.- 686с.
2. Гудман С. Введение в разработку и анализ алгоритмов/ С.Гудман, С. Хидетниеми. – М.: Мир, 1981.-366с.
3. Марченко А.И. Программирование в среде ТурбоПаскаль 7.0/ А.И. Марченко, Л.А. Марченко. - К.: ЮНИОР, 1997. - 496с.
4. Епанешников А.И. Программирование в среде Turbo Pascal 7.0/ А.И. Епанешников, В.А. Епанешников.- М.:"Диалог-МИФИ", 2000.-213с.
5. Прайс Д. Программирование на языке Паскаль. Практическое руководство: Пер. с англ./ Д. Прайс; под ред. Перминова.- М.: Мир,1987.-232 с.
6. . Вирт Н. Алгоритмы и структуры данных/ Н. Вирт.- М.: Мир, 1989.- 360с.
7. Фаронов В.В. Турбо Паскаль (в 3-х книгах). Книга 1. Основы Турбо Паскаля/ В.В. Фаронов.- М.: Учебно-инженерный центр "МВТУ-ФЕСТО ДИАЛЕКТИК", 1992.-304 с.
8. Фигурнов В.Э. IBM для пользователя/В.Э. Фигурнов.- М.: Финансы и статистика, КомпьютерПресс, 1991.-288 с.

ПРИЛОЖЕНИЕ А. Стандартные функции языка Паскаль.

Функция Abs(X) возвращает абсолютное значение аргумента.

X - выражение вещественного или целочисленного типов.

Функция ArcTan(X) возвращает арктангенс аргумента.

X - выражение вещественного или целочисленного типов.

Функция Exp(X) возвращает экспоненту.

X - выражение вещественного или целочисленного типов.

Функция Ln(X) возвращает натуральный логарифм выражения X вещественного или целочисленного типов.

Функция Pi возвращает значение Pi, которое определено как 3.14159265358979323 85.

Функция Sin(X) возвращает синус аргумента в радианах.

X - выражение вещественного или целочисленного типов.

Функция Cos(X) возвращает косинус аргумента в радианах.

X - выражение вещественного или целочисленного типов.

Функция Sqr(X) возвращает квадрат аргумента.

X - выражение вещественного или целочисленного типов.

Функция Sqrt(X) возвращает квадратный корень аргумента.

X - выражение вещественного или целочисленного типов.

ПРИЛОЖЕНИЕ Б. Краткие сведения по работе с оболочкой Norton Commander

Программа Norton Commander(Volkov Commander) является одной из наиболее популярных программ-оболочек для работы с дисковой операционной системой DOS. С ее помощью пользователи могут просматривать каталоги, копировать, переименовывать, удалять файлы , запускать программы и т. д.

Запуск Norton Commander(Volkov Commander) осуществляется набором в командной строке:
NC или VC .

После запуска оболочки на экране появляются два прямоугольных окна, ограниченных двойной рамкой (две панели). Ниже панелей располагается обычное приглашение DOS. Здесь можно вводить команды DOS. Ниже расположена строка-подсказка о назначении функциональных клавиш Norton Commander.

Для выхода из Norton Commander надо нажать клавишу [F10]. На экране появится запрос о подтверждении выхода.

Для получения помощи необходимо нажать клавишу [F1].

Для выполнения программы или команды DOS, необходимо указать ее в командной строке и нажать [Enter].

Переход на другую панель осуществляется нажатием [Tab].

Чтобы выполнить переход в другой каталог надо выделить этот каталог и нажать [Enter]. Переход в корневой каталог осуществляется нажатием [Ctrl-\], переход в надкаталог - [Ctrl-PgUp].

Программа Norton Commander позволяет работать с несколькими файлами, объединенными в группу.

Выбор группы файлов

Включить файл в группу - [Ins].
Исключить файл из группы - [Ins].
Включить в группу файлы по маске - нажать [+] на функциональной клавиатуре и ввести маску.
Исключить из группы файлы по маске - нажать [-] на функциональной клавиатуре и ввести маску.
Выбранные файлы изображаются на цветном дисплее желтым цветом.
Выбранную группу файлов можно:
[F5] - скопировать;
[F6] - переименовать или переместить в другой каталог;
[F8] - удалить.

Управление панелями Norton Commander

[Ctrl-O] - убрать панели с экрана / вывести панели на экран;
[Ctrl-P] - убрать одну из панелей (не текущую) с экрана / вывести панель на экран;
[Ctrl-U] - поменять панели местами;
[Ctrl-F1] - убрать левую панель с экрана / вывести левую панель на экран;
[Ctrl-F2] - убрать правую панель с экрана / вывести правую панель на экран;
[Alt-F1] - вывести в левой панели оглавление другого диска;
[Alt-F2] - вывести в правой панели оглавдение другого диска.

Назначение функциональных клавиш

[F1] - (Help) - получение справки (помощь) ;
[F2] - (User Menu) - вывод меню команд пользователя;
[F3] - (View) –просмотр выделенного файла;
[F4] - (Edit) - редактирование выделенного файла;
[F5] - (Copy) - копирование выделенного файла;

[F6] - (RenMov) - переименование выделенного файла (файлов) или каталога, пересылка файла (файлов) в другой каталог;

[F7] - (MkDir) - создание подкаталога;

[F8] - (Delete) - уничтожение выделенного файла, группы файлов или каталога ;

[F9] - (PullDn) - меню Norton Commander;

[F10] - (Quit) - выход из Norton Commander;

[Shift -F3] - (View) - просмотр файла. Имя файла запрашивается;

[Shift - F4] - (Edit) - редактирование файла. Имя файла запрашивается;

[Shift -F5] - (Copy) - копирование файла или группы файлов. Запрашивается, какие файлы и куда копировать;

[Shift -F6] - (RenMov) - переименование файла (файлов) или каталога,(пересылка файла (файлов) в другой каталог). Запрашивается, какие файлы и как (куда) переименовывать или пересылать;

[Shift -F9] - сохранение текущих режимов Norton Commander;

[Alt -F3] - (View) - просмотр файла с помощью встроенной программы просмотра Norton Commander;

[Alt -F4] - (Edit) - редактирование файла с помощью альтернативного редактора;

[Alt -F7] - (Search) - поиск файла на диске;

[Alt - F8] - (History) - просмотр и повторное выполнение ранее введенных команд;

[Alt - F9] - (EgaLn) - переключение режима отображения количества строк на экране с 25 на 43 или с 43 на 25;

[Alt - F10] - (Tree) - быстрый переход в другой каталог.

Заказ №_____от «___»_____200 ____Тираж_____экз.
Изд-во СевНТУ