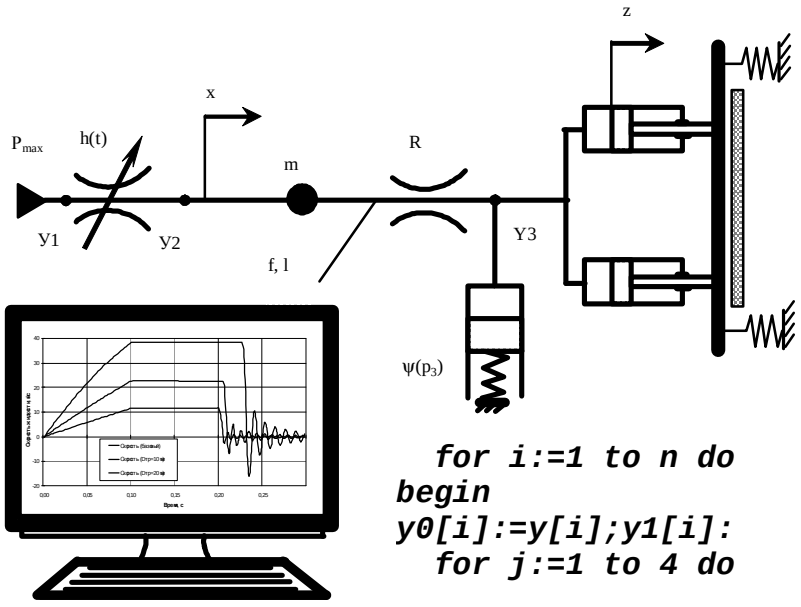
Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра "Гидропневмоавтоматика и гидропневмопривод"

М.И. Жилевич
Л.Г. Филипова

ИНФОРМАТИКА

Учебно-методическое пособие к выполнению лабораторных работ




```
  for i:=1 to n do
begin
y0[i]:=y[i];y1[i]:
  for j:=1 to 4 do
```

М и н с к - 2 0 0 3

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра "Гидропневмоавтоматика и гидропневмопривод"

М.И. Жилевич
Л.Г. Филипова

ИНФОРМАТИКА

Учебно-методическое пособие к выполнению лабораторных работ
для студентов специальности 1-36 01 07
"Гидропневмосистемы мобильных и технологических машин"

Минск 2003

УДК 621.113:681.3(075.8)
ББК 39.33
Ж 72

Жилевич М.И., Филипова Л.Г. Информатика: Учеб.-метод. пособие к выполнению лабораторных работ для студентов специальности 1-36 01 07 "Гидропневмосистемы мобильных и технологических машин".- Мн.: БНТУ, 2003. - с.

Данное пособие предназначено для выполнения лабораторных работ по дисциплине «Информатика» и содержит необходимые теоретические сведения и практические рекомендации, позволяющие программировать на ЭВМ различные вычислительные процессы на языке программирования Паскаль.

Пособие может быть полезно студентам для выполнения расчетов в ходе курсового и дипломного проектирования.

Рецензент: Таяновский Г.А.

ISBN 985-479-003-7

© Жилевич М.И., Филипова Л.Г., 2002

ВВЕДЕНИЕ

Одним из существенных путей повышения эффективности проектирования и эксплуатации гидравлического и пневматического оборудования в настоящее время является использование персональных ЭВМ. Причем ЭВМ применяются не только на стадии научных исследований, но и на этапах проектно-конструкторских работ, для оформления технической документации, сбора и анализа статистической информации при испытаниях, эксплуатации и диагностировании гидро- и пневмоприводов.

Целью курса "Информатика" является обучение студентов методике постановки, подготовки и решения задач по расчету и проектированию гидро- и пневмоприводов на современных ПЭВМ.

В результате изучения дисциплины студенты должны знать:

- общее устройство и характеристики ПЭВМ;
- наиболее распространенные программные оболочки;
- один из алгоритмических языков;
- сущность численных методов, используемых при расчетах гидро- и пневмоприводов;
- возможности применения ЭВМ и микропроцессорной техники в области проектирования, испытаний и эксплуатации объектов с гидравлическим и пневматическим приводом.

Студент должен уметь:

- работать на ПЭВМ в качестве пользователя;
- создавать и редактировать текстовые и графические документы;
- ставить задачу и разрабатывать алгоритм ее решения;
- составлять программы и выполнять расчеты гидропневмоприводов и их элементов;
- применять численные методы для исследования процессов в гидропневмосистемах.

Дисциплина "Информационные технологии и программирование" является общетехнической, на ней базируется изучение специальных дисциплин, таких как "Математическое моделирование производственных процессов", "Автоматизированное проектирование гидропневмоприводов", "Теория и проектирование гидропневмосистем", "Техническая диагностика гидропневмосистем" и др. Расчеты на ЭВМ выполняются в ходе курсового и дипломного проектирования.

В ходе выполнения каждой из лабораторных работ студент должен оформить *отчет*, содержащий следующие разделы:

- цель работы;
- задание и исходные данные;
- схема алгоритма решения задачи (головной программы и подпрограмм);
- таблица идентификаторов;
- распечатка головной программы, подпрограмм и результатов расчета;
- выводы.

ЛАБОРАТОРНАЯ РАБОТА №1

Изучение основных функций пакета NORTON COMMANDER и команд операционной системы MS DOS

Цель работы:

1) *Ознакомиться с основными возможностями пакета программ NORTON COMMANDER и некоторыми командами операционной системы MS DOS.*
2) *Приобрести навыки работы в среде NORTON COMMANDER .*

NORTON COMMANDER (NC)- пакет прикладных программ с сервисными функциями (оболочка операционной системы).

NC предназначен для облегчения взаимодействия пользователя с операционной системой MS DOS.

Этот пакет позволяет взаимодействовать с MS DOS при помощи:

1) курсора;
2) функциональных клавиш;
3) манипулятора типа “мышь”;
4) обычных команд операционной системы.

Запуск пакета осуществляется командой NC. После этого на экране появится изображение одной или двух панелей. На панели может быть информация:

1) оглавление каталога на диске (как правило эта информация основная и вводится без дополнительных команд);
2) дерево каталога на диске;
3) сводная информация о диске и каталоге, изображаемом на противоположной панели.

В нижней строке экрана выводится напоминание о назначении функциональных клавиш:

F1- краткая информация о назначении клавиш при работе с NC;

F2- пользовательское меню (спец. перечень команд пользователя);

F3- просмотр текстового файла без изменения;

F4- редактирование файла;

F5- копирование файла.

По этой команде высвечивается запрос о том, куда копировать файл. По умолчанию файл копируется на диск или в каталог расположенный в противоположной панели. Для копирования файла в другой каталог необходимо набрать его маршрут. Для отмены команды - нажать клавишу ESC;

F6- переименование файла;

F7- создание подкаталога;

F8- удаление файла из подкаталога;

F9- в верхней строке экрана выводится меню, содержащие информацию о режимах работы NC. Используются для настройки NC;

F10- выход из NC.

Выбор группы файлов осуществляется подсвечиванием этого файла и нажатием клавиши вставка (INS). При этом буквы выделенного файла имеют повышенную яркость. Для отмены выделения используется клавиша вставка.

Ниже приводится назначение некоторых основных сочетаний клавиш.

Клавиша табуляция (TAB)- переход с левой на правую панель.

CTRL+F1- закрыть, открыть левую панель.

CTRL+F2- закрыть, открыть правую панель.

CTRL+O- закрыть, открыть обе панели.

CTRL+U- поменять местами панели.

ALT+F1- для перехода на другой диск с левой панели.

ALT+F2- для перехода на другой диск с правой панели.

(SHIFT)+F4- создать новый файл.

ALT+F10- вывод на экран дерева подкаталогов на диске.

CTRL+E- вывод предыдущей выполненной команды в командную строку.

CTRL+X- вывод последующей выполненной команды.

ALT+F7- быстрый поиск файла на диске.

CTRL+F8- вывод списка введенных команд.

ALT+F8- просмотр и выполнение ранее введенных команд.

ЗАДАНИЕ.

1. Запустите пакет NORTON COMMANDER.
   Действие: Набрать NC и нажать ВВОД (либо воспользоваться ярлыком на рабочем столе).
   Результат: на экране появится 2 окна.
2. Апробировать клавиатуру
   Действие: Нажать клавишу табуляции TAB.
   Результат: Перемещение курсора из одного окна в другое.
3. Апробировать работу всех клавиш управления курсором.
   Действие: Поочередно нажать клавиши управления курсором. Результат: Перемещение курсора по названиям файлов в пределах одного окна.
4. Перейти в левое окно.
   Действие: Нажать клавишу TAB.
   Результат: Перемещение курсора в левое окно.
5. Загрузить в левое окно каталог диска D.
   Действие: Набрать с клавиатуры D: и нажать ВВОД.
   Результат: На верхней границе окна появится надпись D:\

6. Перейти на правое окно.
   Действие: Нажать клавишу TAB.
   Результат: Курсор-подсветка перемещается в правое окно.
7. Загрузить в правое окно каталог диска C.
   Действие: C: ВВОД.
   Результат: На верхней границе появится надпись C:\
8. Перейти в левое окно.
9. Создать каталог с именем PROB.
   Действие: MC PROB ВВОД.
   Результат: В левом окне появляется имя созданного каталога.

10. Войти в каталог PROB.
    Действие: CD PROB ВВОД.
    Результат: В верхней части левого окна появится надпись D:\PROB

11. Создать в каталоге PROB файл prob1.txt
    Действие: SHIFT+F4, в ответ на запрос необходимо набрать имя файла.
    Результат: Загружается текстовый редактор и окно для набора текста.
    Действие: Набрать какую-либо информацию, нажав две-три цифровых клавиши, а затем нажать клавишу F10.
    Результат: Редактор предлагает три альтернативы:
        1. SAVE (сохранить информацию);
        2. DON'T SAVE (не сохранять);
        3. CONTINUE EDITING (продолжить редактирование).
    Действие: С помощью клавиш управления курсором подвести курсор-подсветку на SAVE и нажать ВВОД.
    Результат: В каталоге PROB появляется файл с именем prob1.txt

12. Удалить файл prob1.txt
    Действие: DEL prob1.txt и клавиша ВВОД.
    Результат: Имя файла исчезает из каталога.

13.Перейти в корневой каталог D.
    Действие: CD\ и клавиша ВВОД.
    Результат: В верхней части левого окна появляется маршрут к каталогу D. В окне располагается список файлов D.

14. Удалить каталог PROB.
    Действие: RD PROB и клавиша ВВОД.
    Результат: Имя каталога исчезает из оглавления диска D.

15. Войти в каталог USERS (при необходимости создать).

Действие: Совместить курсор с именем каталога и нажать клавишу ВВОД.

Результат: В левом окне выводится список подкаталогов и файлов каталога USERS.

16. Создать персональный каталог со своим именем, например MY1.

Действие: F7.

Результат: Появляется запрос об имени каталога.

Действие: Набрать MY1 и нажать ВВОД.

Результат: В списке каталога USERS появляется подкаталог MY1.

17. Войти в персональный каталог.

Действие: см п. 15.

18. Создать в персональном подкаталоге файл LAB1.TXT.

Действие: см. п. 11.

19. С помощью клавиш управления установить курсор-подсветку на файл LAB1.TXT.

20. Войти в режим редактирования.

Действие: F4.

Результат: Файл LAB1.TXT выводится для редактирования.

21. Набрать предложенную преподавателем информацию.

22. Обеспечить запись набранной информации в память машины без выхода из текстового редактора.

Действие: F2.

Результат: Выводится сообщение о сохранении файла. После этого можно продолжать редактирование текста.

23. Закончить работу с редактором текста.

Действие: ESC или F10.

Результат: Редактор предлагает три альтернативы:

1. SAVE (сохранить информацию);
4. DON'T SAVE (не сохранять);
5. CONT.EDITING (продолжить редактирование).

Примечание: Если после предыдущего сохранения информации в редактируемый файл изменения не вносились, после нажатия на ESC или F10 осуществляется выход из текстового редактора.

Действие: С помощью клавиш управления курсором подвести курсор-подсветку на . SAVE и нажать клавишу ВВОД.

24. Перейти на правое окно (панель) в каталог USERS диска D.

Действие: см. п.п.4, 5, 15.

Результат: см. п.15.

25. Вернуться на левое окно.
26. Подставить файл LAB1.TXT.

27. Скопировать файл LAB1.TXT в каталог USERS.
Действие: F5.
Результат: Высвечивается запрос о том, куда копировать файл.
Действие: Напечатать имя носителя D:/ USERS и нажать ВВОД.
Результат: На правой панели, отражающей состояние каталога USERS, появляется копия файла LAB1.
28. Перейти на правое окно.
29. Подсветить файл LAB1.
30. Просмотреть его без редактирования.
Действие: F3.
Результат: На экран выводится текст, просмотреть его.
Действие: ESC.
Результат: Возврат в окно.
31. Создать файлы LAB2.TXT и LAB.TXT.
Действие: см. п. 11.
32. Проверить срабатывание ошибочно (непреднамеренно) набранной команды.
Действие: Подсветить файл LAB2.TXT и нажать клавишу F8 (удаление).
Результат: Появляется запрос об удалении.
Действие: ESC.
Результат: Выход в основное окно.

33. Отметить группу файлов для удаления (LAB1.TXT, LAB2.TXT, LAB.TXT).
Действие: Подсветить первый файл и выделить его, нажав клавишу вставки.
Результат: Буквы составляющие имя файла будут выделены ярким цветом.
Действие: Подсветить следующий файл и выделить его и т. д.
Результат: Все выделенные состоят из ярких букв.

34. Удалить выделенную группу файлов.
Действие: F8.
Результат: Все выделенные файлы исчезают из каталога.

35. Завершить работу с пакетом NORTON COMMANDER.
Действие: F10.
Результат: Запрос на подтверждение команды.

Действие: С помощью клавиш управления курсором совместить подсвеченный прямоугольник с YES (или NO) и нажать ВВОД.

Осуществляется выход в ОС. Для возврата в NORTON COMMANDER набрать NC и нажать ВВОД.

36. Распечатать файл.

Действие: Перейти в соответствующий каталог и подсветить требуемый файл, а затем нажать F5.

Результат: Высвечивается запрос о том, куда копировать файл.

Действие: PRN и ВВОД (предварительно подготовив принтер).

37. Закрыть (открыть) левую панель.

Действие: CTR+F1.

Результат: Изображение левой панели исчезает.

Действие: CTR+F1.

Результат: Изображение восстанавливается.

38. Закрыть (открыть) правую панель.

Аналогично п. 37: CTR+F2.

39. Закрыть (открыть) обе панели.

Аналогично п. 37: CTR+O.

40. Поменять местами панели.

Действие: CTR+U.

41. Перейти в другой подкаталог через дерево каталогов.

Действие: ALT+F10.

Результат: На панели отображается дерево каталогов на диске.

Действие: Подсветить выбранный каталог и нажать ВВОД.

Результат: На панели отображается окно с содержимым выбранного каталога.

42. Осуществить обратный переход в персональный каталог (аналогично п.41).

43. Осуществить переход на другой диск(C).

Действие: ALT+F2.

Результат: На правой панели выводится список дисков, имя одного подсвечено.

Действие: С помощью курсора подсветить требуемый диск (C) и нажать ВВОД.

Результат: На правой панели отображается содержание диска (C).

44. Осуществить возврат на рабочий диск.

Действие: ALT+F1 (аналогично п. 43, но на правой панели).

45. Удалить все созданные пробные каталоги и файлы в результате выполнения лабораторной работы.

ЛАБОРАТОРНАЯ РАБОТА №2

Структура программы на языке Паскаль и порядок работы в системе программирования TURBO PASCAL 7.0

Цель работы:

1) *Изучение структуры программы на языке Паскаль.*
2) *Изучение основных положений среды программирования TURBO PASCAL 7.0.*
3) *Приобретение навыков работы в среде программирования TURBO PASCAL 7.0.*

Программа на языке Pascal *состоит* из заголовка и собственно программы, называемой блоком. Блок может содержать другие блоки. Блок состоит из двух частей: описательной и исполнительной. Блок, который не входит ни в какой другой, называется *глобальным*. Если в глобальном блоке находятся другие блоки, они называются *локальными*. Глобальный блок - это основная программа. Он должен присутствовать в любом случае. Локальные блоки - это процедуры и функции (они могут отсутствовать).

Объекты программы (типы, переменные, константы и т.д.) соответственно называются глобальными и локальными. Область действия объектов - тот блок, где они описаны, и все вложенные в него блоки.

В начале программы находится *заголовок*, состоящий из слова PROGRAM, имени программы и точки с запятой.

Например:

```
PROGRAM LAB1;
```

После заголовка можно расположить *комментарий*, в котором содержится текстовая информация о назначении программы, разработчике и т.д. Комментарии могут быть вставлены в любое место программы, при этом её смысл не изменяется. Комментарий заключается в фигурные скобки или в символы (* *).

Например:

{лабораторная работа N1} или (*Выполнил Иванов*).

Непосредственно *программный блок* состоит из следующих *разделов*:

```
USES имя1,имя2,...; -раздел подключаемых библиотечных модулей;
LABEL ...;    - раздел меток;
CONST ...;    - раздел констант;
TYPE ...;     - раздел определения типов данных;
VAR  ...;     - раздел описания переменных;
PROCEDURE имя; - раздел описания процедур;
  (тело процедуры)
FUNCTION  имя; - раздел описания функций;
  (тело функции)
BEGIN   (операторы) - раздел операторов;    END.
```

Любой из разделов, кроме раздела операторов основной программы, может отсутствовать. Разделителем между разделами и операторами является точка с запятой. В конце программы должна стоять точка. Раздел операторов заключается в операторные скобки BEGIN ... END. В этом разделе указывается последовательность действий, которые должна выполнить ЭВМ.

Раздел USES состоит из слова USES и списка имён подключаемых стандартных и пользовательских библиотечных *модулей*. Например, USES Graf; {подключение графического пакета}.

Раздел LABEL используется для описания так называемых *меток*. Перед любым оператором можно поставить метку, что позволяет выполнить переход на этот оператор из любого места блока с помощью оператора GOTO. Метка состоит из имени и следующего за ним двоеточия. Именем может служить идентификатор или цифра. Максимальная длина имени метки - до 127 символов. Перед употреблением метка должна быть описана. Раздел описания начинается словом LABEL, за которым через запятую записываются имена меток, а за последним именем ставится точка с запятой. После записи метки в разделе операторов следует двоеточие.

Например:

```
LABEL M1,M2;
  BEGIN
    ...
  M1: оператор;
  M2: оператор;
    ...
  END.
```

В *разделе описания констант* производится присвоение идентификатором констант постоянных значений. Раздел начинается словом CONST, после которого следует ряд выражений, присваивающих идентификаторам постоянные числовые или строковые значения.

Например: CONST Tel1='Иванов'; PI=3.1415; .

Раздел описания типов данных начинается словом TYPE, за которым следуют одно или несколько определений типов, разделённых точкой с запятой.

Например: TYPE Mas=array[1..10] of real; Dni=1..31; .

Раздел описания переменных начинается словом VAR. Затем через запятую перечисляются имена переменных, а через двоеточие следует их тип, причем в конце описания каждого типа ставится точка с запятой.

Например: VAR A,B,C:integer; TEV, S:real; L :boolean;.

В разделе описания процедур и функций размещаются подпрограммы. В общем случае подпрограмма имеет ту же структуру, что и программа. Общая форма записи процедур и функций:

```
PROCEDURE имя процедуры (параметры);
  раздел описаний;
 BEGIN
  раздел операторов
 END;

FUNCTION имя функции (параметры): тип результата;
  раздел описаний;
BEGIN
  раздел операторов;
END;
```

Параметры в процедуре и функции заключаются в скобки и перечисляются через запятую с указанием их типа, параметры различных типов отделяются точкой с запятой. Однако параметры могут и отсутствовать.

Среда программирования Turbo Pascal представляет собой интерактивную среду, включающую экранный редактор, компилятор, редактор связей и отладчик. Интегрированная среда позволяет набирать тексты программ с использованием встроенного редактора текстов, компилировать их, выполнять, проводить отладку программ.

Загрузка системы Turbo Pascal осуществляется при помощи запуска файла Turbo.exe, который находится в директории пакета Turbo Pascal (TP). После выполнения загрузки на экране дисплея появляется основной экран интегрированной среды (рис. 1), состоящий из трех частей: строки главного меню, поля экрана, строки состояния.

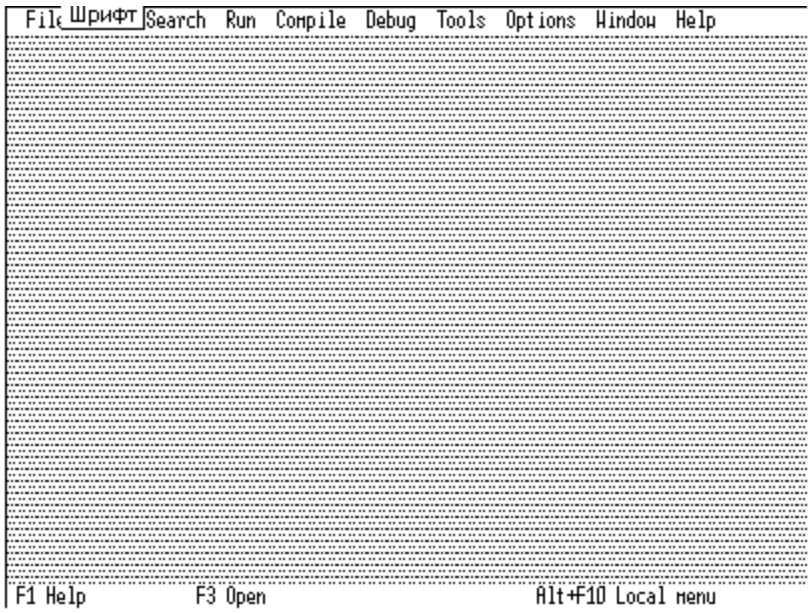


Рис.1. Основной экран

Правая строка содержит все команды главного меню. Вход в главное меню осуществляется с помощью функциональной клавиши F10. В последней строке экрана приведены основные доступные на каждый текущий момент функциональные клавиши с указанием их назначения. Содержимое строки состояния меняется при изменении режима работы среды.

Главное меню системы обеспечивает следующие режимы работы:

File - команды управления файлами;

Edit - команды текстового редактора для работы с блоками;

Search - команды поиска нужного текста или ошибок в файле;

Run - команды выполнения или пошагового выполнения программы;

Compile – компиляция программы на диск или в память;

Debug - команды отладки программы (оценивать выражения, изменять данные, устанавливать точки прерывания и окно просмотра значений переменных);

Tools – команды трассировки и инструментальные программные средства пользователя;

Options - установка и изменение режимов работы для компилятора, редактора, мыши, отладчика и т. д.

Window - команды работы с окнами (открывать, размещать, просматривать);

Help – вызов информации о работе системы(помощь);

В данной лабораторной работе полностью или частично описаны локальные меню следующих режимов работы: File, Edit, Run, Compile, Debug, Window.

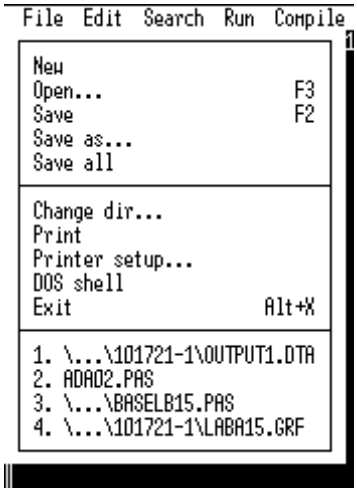


Рис.2. Локальное меню режима File

Локальное меню режима File приведено на рис.2 и содержит следующие команды:

New – создать новый файл в новом окне редактирования. При выборе этой команды на экране появляется окно редактирования (рис.3)

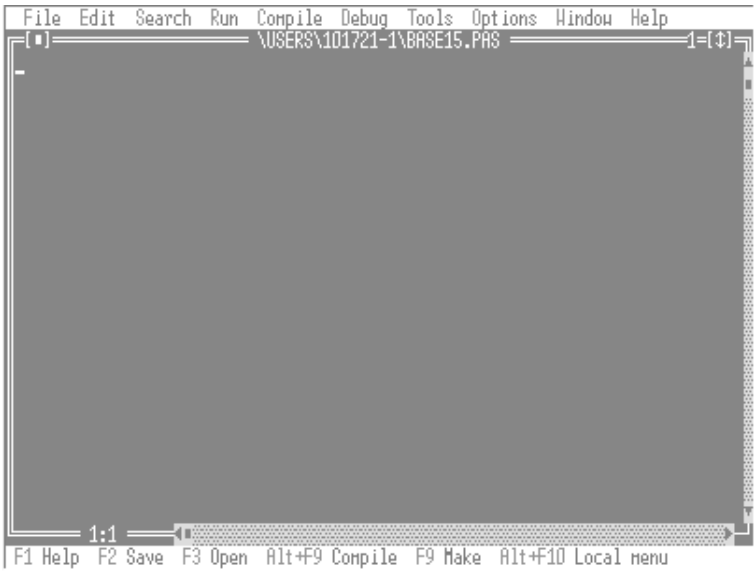


Рис.3. Окно редактирования

Имя файла в даном случае Noname00. pas, так как файл еще не записан на диск с определенным именем.

Open (F3) – обнаруживать и открывать файл в окне редактирования. Выбор этой команды ведет за собой появления на экране диалогового окна “Open a file” (рис.4).

Рис.4. Диалогового окно “Open a file”

В этом окне имеется несколько рабочих зон: 1 – заголовок меню; 2 – строка ввода имени нужного файла; 3 – содержание открытого в данный момент каталога; 4 – строка состояния; 5 – открытый файл с именем, набранным в строке 2, в новом окне редактирования; 6 – загрузить в окно редактирования файл, выбранный курсором в активном окне 3; 7 – закрыть диалоговое окно без изменений; 8 – просмотр помощи о работе в диалоговом окне.

При открытии диалогового окна курсор ожидает ввода имени файла в строке 2. Если необходимо провести поиск нужного файла на диске, нажмите “Tab” и окно 3 станет активным. Перемещение курсора в окне 3 осуществляется при помощи клавиш со стрелками. При подведении курсора к очередному имени файла в строке состояния 4 появляется информация о месте расположения этого файла, его размерности, дате и времени создания.

Для подтверждения ввода имени файла или выбора его на диске нажмите клавишу “Enter”, либо при помощи клавиши “Tab” перейдите к окнам 5 или 6. Для выхода из диалогового окна нажмите клавишу “Esc” либо перейдите к окну 7.

Save(F2) - сохранить файл, находящийся в активном окне редактирования.

Save as – сохранить находящийся в активном окне редактирования файл под другим именем либо перенести его на другой каталог, или на другой диск. При выборе этой команды на экране появляется диалоговое окно “Save file as”. Работа в нем не отличается от описанной выше работы в диалоговом окне “Open file as” (рис.4)

Change dir - перейти в другой каталог или на другой диск.

Printer setup - задать фильтр для вывода текста на принтер, тип принтера и возможность выделения шрифтами различных элементов программы.

DOS shell - временный выход в DOS.

Exit - выход из системы Turbo Pascal.

Локальное меню режима **Edit** приведено на рис.5 и содержит следующие команды:

Рис.5. Локальное меню режима Edit

Undo - откат, отмена выполнения эффекта предыдущей команды или группы команд и восстановления состояния обрабатываемого текста.

Redo - возврат, команда противоположная Undo.

Cut - удалить выделенный блок текста из программы и поместить его в буфер обмена clipboard.

Copy - копировать выделенный блок текста без удаления его из текущей программы в буфер обмена.

Paste- вставить блок текста из буфера обмена в текущий файл.

Clear - удалить выделенный блок текста без записи его в буфер обмена.

Show clipboard - открыть окно буфера обмена.

Локальное меню режима Run приведено на рис.6 и содержит следующие команды:

Run - выполнить программу, находящуюся в активном окне редактирования.

Setup over - выполнить следующий оператор программы без ухода внутрь программы.

Trace into - выполнить следующий оператор программы с возможностью выполнения операторов внутри подпрограммы.

Go to cursor - начать выполнение программы с оператора, на котором находиться курсор.

Program reset - прервать сеанс отладки программы и освободить память.

Parameters - задать параметры программе точно так же, как они задаются при запуске программы с помощью командной строки.

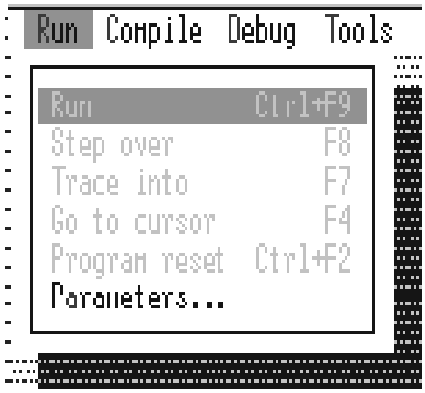


Рис.6. Локальное меню режима Run

Локальное меню режима **Compile** приведено на рис.7 и содержит следующие команды:

Compile(Alt + F9) - трансляция только программы, указанной в primary file (или находящейся в редактор, если в primary file отсутсвует имя файла).

Make - трансляция программы, указанной в primary file или находящейся в редакторе, и перетрансляция всех модулей пользователя, подсоединеных к данной программе, в которую были внесены изменения.

Build - трансляция программы, указанной в primary file или находящейся в редакторе, и перетрансляция всех модулей пользователя, подсоеди-

ненных к данной программе, вне зависимости от того были ли внесены в них изменения.

Destination – определяет, где будет сохраняться построенный EXE-файл: в памяти или на диске.

Primary file - определяет файл, который будет компилироваться. Если файл не указан, то компилируется файл, загруженный для редактирования.

Clear primary file - выгружает файл, предварительно посланный в primary file.

Information - вывод информации о текущем каталоге или файле, программного кода, кода завершения программы, размере стека и данных.

Рис.7. Меню режима Compile

Локальное меню **Debug** приведено на рис.8 и содержит следующие команды**:**

Breakepoints - установить условные точки прерывания программы, просмотреть и редактировать их. Точка прерывания - контрольная точка программы, в которой ее выполнение прерывается и ее выполнение передается отладчику.

Call stack - открыть окно обращения к процедурам и функциям в текущей точке программы с целью просмотра значений параметров выделенной процедуры или функции.

Register - открыть окно регистров и их разрядности.

Watch - открыть окно просмотра текущих значений выбранных переменных или выражения программы.

Outptut - Открыть окно отображения данных. Окно Outptut показывает текст любой последней команды DOS или текст, сгенерированный из вашей программы.

User screen - просмотр полного экрана пользователя(окно Outptut в полноэкранном режиме).

Evaluate/modify - команда дает возможность задать переменную или

выражение, для которого следует вычислить значение или задать новое значение для переменной.

Add watch - вставить наблюдаемое выражение или переменную в окно Watch.

Add Breakpoint - определить новую точку прерывания.

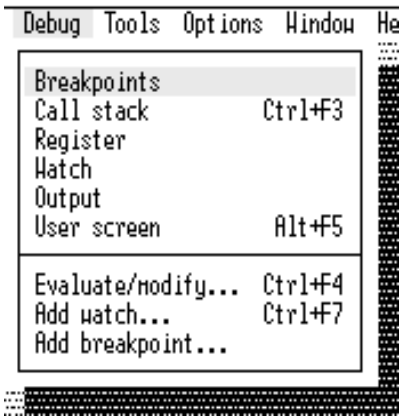


Рис.8. Локальное меню Debug

Рис.9. Локальное меню Window

Локальное меню **Window** приведено на рис.9 и содержит следующие команды:

Tile - упорядочить все открытые окна в виде непересекающихся окон.

Cascade - упорядочить все открытые окна на рабочем окне в каскадном порядке.

Close all - закрыть все открытые окна.

Refresh display - комманда восстановления экрана.

Size/move - изменить размеры и позицию на экране активного окна.

Zoom - увеличить или восстановить размеры активного окна.

Next - сделать следующее окно активным.

Previous - сделать предыдущее окно активным.

Close - закрыть активное окно.

List - показать список всех открытых окон.

ЗАДАНИЕ 1

1. Войти в интегрированную среду Turbo Pascal 7.0
2. Войти в окно редактирования (F10/File).
3. Сменить каталог на D:\users\Gr(.....номер вашей группы) (Change dir).
4. Создать новый файл выполнения программы по заданию преподавателя. (New).
5. Набрать программу, предложенную преподавателем.
6. Вызвать компилятор(Compile/Compile).
7. Исправить ошибки.
8. Запустить выполнение программы (Run/Run).
9. Просмотреть результаты выполнения программы на пользовательском экране(Debug/User screen).
10. Сохранить выполненную программу(File/Save).
11. Распечатать программу(File/Print).
12. Сохранить выполненную программу под другим именем(File/Save as).

ЗАДАНИЕ 2

1. Открыть созданный программный файл (File/Open).
2. Выделить блок текста программы (Shift/).
3. Удалить выделенный блок текста из программы в карман (Edit/Cut).
4. Вставить блок текста из кармана в место, указанное курсором (Edit/Paste).
5. Копировать вставленный блок текста на старое место (Edit/Copy).
6. Удалить выделенный блок текста (Edit/Clear).
7. Отменить предыдущую команду (Edit/Undo).
8. Возвратить предыдущую команду (Edit/Redo).
9. Открыть окно кармана (Edit/Show Clipboard).
10. Получить информацию о программном файле (Compile/Information).
11. Просмотреть другие команды режимов (Debug, Window).

ЛАБОРАТОРНАЯ РАБОТА №3

Ввод-вывод различных типов данных

Цель работы:

1) Изучение стандартных типов данных.

2) Ознакомление с правилами описания различных типов данных.

3) Изучение способов ввода и вывода различных типов данных.

Ввод данных можно осуществить следующими способами:

1) значения задаются непосредственно в программе в разделе констант или в тексте программы операторами присваивания;

2) с помощью операторов ввода (с клавиатуры или через внешний файл).

Для ввода данных используются операторы:

READ (список); READLN (список);

причем в списке переменных через запятую перечисляются идентификаторы вводимых переменных (список может отсутствовать).

При вводе данных (например, с клавиатуры) каждое вводимое значение последовательно присваивается переменным из списка. Разделителями между числами служат пробел или нажатие клавиши ввод. В данной лабораторной работе будем рассматривать организацию ввода данных с клавиатуры, вывода - на дисплей.

Особенностью действия оператора READLN является то, что после завершения вода данных осуществляется переход на новую строку.

Например,

READ(a,b,d); - ввод переменных a, b, d без перехода на новую строку;

READLN(a,b); READ(t); - переменные вводятся в две строки, после считывания переменных a, b курсор перемещается на новую строчку, откуда считывается переменная t.

Оператор ввода без списка READLN; используется для задержки выполнения программы, программа запускается после нажатия клавиши ввода.

Необходимо обратить внимание, что при вводе символьной информации разделитель между символами не нужен (так как под символом подразумевается только один знак). Кроме того, перед вводом символьных переменных необходимо записать оператор Readln, так как при загрузке файла ввода автоматически загружается пустая строка из пробела, который при считывании цифровой информации воспринимается как разделитель и игнорируется, а при вводе символьной - этот пробел необходимо прочитать (он воспринимается как символ).

Например: var x:char;

```
begin  readln;  read(x);  end.
```

Для вывода данных используются операторы WRITE и WRITELN.

Общая форма записи операторов:

WRITE (список); WRITELN (список);

причем в списке через запятую перечисляются выводимые переменные (список может отсутствовать). Оператор выводит последовательно значения переменных из списка.

Особенностью действия оператора WRITELN является то, что после завершения вывода данных осуществляется переход на новую строку (следующий оператор вывода будет выводить данные с начала новой строки).

Например,

WRITE(a,b); WRITE(t); - значения переменные a,b,t выводятся подряд (в одну строку);

WRITELN(a,b); WRITE(t); - значения переменных выводятся в две строки, после вывода значений переменных a, b курсор перемещается на новую строчку и выводит значение переменной t;

WRITELN; - оператор без списка используется для перехода на новую строку без вывода данных (можно организовать вывод пустой строки).

В списке выводимых данных могут быть также выражения, символьные константы и переменные. Символьные данные в операторе вывода заключаются в апострофы (') и отделяются запятой. Информация в апострофах выводится без изменения, это используется для того, чтобы снабдить результаты выполнения программы соответствующими надписями. Например, оператор

WRITELN('РЕЗУЛЬТАТ=', a+b/c);

осуществит вывод следующей информации (если значение выражения a+b/c равно 3): РЕЗУЛЬТАТ=3.

При выводе логической переменной печатается ее значение. Например, результатом действия оператора Writeln(a<7) будет вывод слова TRUE, если a=1, и вывод слова FALSE, если a=10 .

В Турбо-Паскале предусмотрен форматный вывод данных. Под форматом подразумевается заранее предопределенная структура выводимых данных (например, количество символов в выводимом числе).

Для целых чисел форматный вывод осуществляется оператором

WRITE(A:d);

для вещественных - оператором

WRITE(A:d:w);

где A – идентификатор выводимых данных;

d - количество позиций, отводимых под число;

w - количество позиций, отводимых под дробную часть числа, причем под точку, отделяющую дробную часть числа, отводится отдельная позиция.

Если количество выводимых символов меньше d , то запись числа дополняется пробелами слева.

Например, если A=1, то по команде WRITE(‘A=’, A:3); выведется запись A=..1 (здесь точка означает пробел, или пустой символ).

Если формат не указан, вывод осуществляется в стандартной форме. Если для вещественных чисел указано только d , число выводится в экспо-

ненциальной форме с шириной поля d. При выводе символьных переменных, каждому символу ставится в соответствие одна переменная. Количество позиций под символьную переменную может задаваться переменной целого типа. Это позволяет строить графики. Например, WRITELN('*':k); где k- вычисляется в программе и определяет положение звездочки относительно некоторой нулевой линии.

ЗАДАНИЕ.

Разработать программу для ввода-вывода различных типов данных в соответствии с заданным вариантом (см. табл.1).

Ввод исходных данных осуществить с клавиатуры, предусмотрев диалоговый режим работы. Результаты расчета вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть оистку экрана и задержку выполнения программы в конце расчета.

Таблица 1

Варианты заданий

Номер варианта	Целые числа.			Вещественные числа.				Количество символов	Вывести значение булевской переменной
	Количество	Формат		Количество	Формат				
		стандарт-ный	ширина поля		стандарт-ный	ширина поля	количество знаков после точки		
1	2	+	5	4	+	6	2	4	TRUE
2	3	+	6	2	+	7	3	5	FALSE
3	3	+	6	2	+	6	3	3	TRUE
4	2	+	5	4	+	6	2	6	TRUE
5	4	+	4	3	+	7	4	4	FALSE
6	2	+	3	3	+	5	1	3	TRUE
7	3	+	2	4	+	8	3	2	TRUE
8	3	+	5	4	+	6	2	5	FALSE
9	4	+	7	3	+	5	1	6	FALSE
10	2	+	6	5	+	8	4	7	TRUE
11	2	+	6	5	+	6	2	6	TRUE
12	3	+	4	6	+	6	2	5	FALSE
13	3	+	3	6	+	5	2	4	FALSE
14	3	+	2	4	+	5	2	3	TRUE
15	4	+	5	5	+	7	3	4	TRUE
16	2	+	7	2	+	7	4	2	FALSE
17	2	+	8	4	+	4	1	4	TRUE
18	3	+	6	4	+	6	2	4	FALSE
19	3	+	6	5	+	0	2	3	FALSE
20	3	+	4	5	+	7	3	3	TRUE

ЛАБОРАТОРНАЯ РАБОТА №4

Программирование линейных алгоритмов. Вычисление выражений с использованием стандартных функций

Цель работы:

1) *Изучение порядка действий при вычислении арифметических выражений на ЭВМ.*
2) *Приобретение навыков в записи выражений на языке Паскаль с использованием стандартных функций.*

Линейным называется алгоритм, в котором результат получается путем однократного выполнения заданной последовательности действий при любых значениях исходных данных. Операторы будут задействованы последовательно, один за другим, в соответствии с их расположением в тексте программы.

Выражение задаёт порядок выполнения действий над элементами данных и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций.

Оператор присваивания тождественен знаку равно в математической записи выражения. Оператор присваивания имеет вид := (двоеточие и равно). При выполнении оператора присваивания вычисляется выражение, стоящее в правой части, и его значение присваивается переменной в левой части. Тип результата выражения должен соответствовать типу переменной в левой части.

Операции подразделяются на арифметические, отношения, логические (булевские), строковые и др.

Арифметические операции выполняют арифметические действия в выражениях над операндами целого и вещественного типов. К основным арифметическим операциям можно отнести следующие: + - сложение; - - вычитание; * - умножение; / - деление; div - целочисленное деление; mod - остаток от целочисленного деления; - - отрицание знака.

Выполнение каждой операции происходит с учётом их приоритета: сначала операции типа умножения (*, /, div, mod), а затем операции типа сложения (=,-). Операции одного и того же старшинства выполняются слева направо в порядке их появления в выражении. Выражения в круглых скобках вычисляются в первую очередь. В выражения могут входить ***стандартные (встроенные) функции***. Их вычисление предшествует выполнению арифметических операций.

Основные встроенные функции: SIN(X) - вычисление sin x; COS(X) - вычисление cos x; LN(X) - вычисление натурального логарифма ln x; SQR(X) - вычисление x^2; ARCTAN(X)- вычисление arctg x; EXP(X)- вычисление e^x;

SQRT(X)- вычисление $\sqrt{x}$; ABS(X)- вычисление $|x|$; TRUNC(X)-вырабатывает целый результат путём отбрасывания дробной части аргумента; ROUND(X)-вырабатывает целый результат путём округления до ближайшего целого; ODD(x) - принимает значение True, если x - нечетное, и False, если x - чётное.

В Паскале определены только три тригонометрические функции (SIN, COS, ARCTAN), для вычисления остальных необходимо использовать известные из тригонометрии соотношения:

Tg(x)=sin(x)/cos(x);

Ctg(x)=cos(x)/sin(x);

Arcsin(x)=Arctg$\left(x/\sqrt{1-x^2} \right)$;

Arccos(x)=π/2-Arcsin(x);

Arcctg(x)=π/2-Arctg(x).

Полезным для написания вычислительных программ является преобразование x^a=EXP(a*LN(x));

При выполнении арифметических операций над величинами только вещественного, а также вещественного и целого типов вырабатывается результат вещественного типа.

ЗАДАНИЕ

Разработать программу вычисления значения выражения в соответствии с заданным вариантом (см. табл.2). Ввод исходных данных осуществить с клавиатуры, предусмотрев диалоговый режим работы. Результаты расчета вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета.

Таблица 2

Варианты заданий

№ варианта	Арифметическое выражение
1	$a = ln(\, y^{\sqrt{\lvert x \rvert}}\,) \times \left(sin(x) + e^{(x+y)} \right)$
2	$b = \sqrt{c \times \left(\sqrt{y} + x^2 \right)} \times \left(cos(x) - \lvert c - y \rvert \right)$
3	$c = arctg(\,x\,) - \frac{3}{5} \times e^{x \times y} + 0.5 \times \frac{\lvert x + y \rvert}{\left(x + y^b \right)}$
4	$d = \frac{e^{\lvert x-y \rvert} \times tg(z)}{arctg(y) + \sqrt{x}} + ln(\,x\,)$

№ варианта	Арифметическое выражение
5	$e = \frac{\left(cos(x) - sin(y)\right)^3}{\sqrt{tg(z)}} + ln^2\left(x \times y \times z\right)$
6	$f = y^x + \sqrt{\lvert x \rvert + e^y} - \frac{z^3 \times sin^2(y)}{y + z^2 / (y - x)}$
7	$g = \frac{1 + cos(x + y)}{\left\lvert e^x - 2 \times y / \left(1 + x^2 \times y^2\right)\right\rvert} \times x^3 + arcsin(y)$
8	$h = 2 + \frac{x^2}{\sqrt{2}} + \frac{\lvert y^3 \rvert}{\sqrt{2}} + \frac{z^4 \times (ln(x) + 1) \times \sqrt{2}}{\sqrt{3}}$
9	$j = \left((1 + y) \times \sqrt{sin^2(z)} - \frac{\lvert y - x \rvert}{5} \right)^3$
10	$k = ln\left\lvert \left(y - \sqrt{x}\right) \times \left(x - \frac{y}{z + x^2 / 4}\right) \right\rvert$
11	$l = 0.5 \times x^5 + 3 \times cos(x + y) + e^{-0.1 \times y \times z} - \sqrt{\lvert x \times y \rvert}$
12	$m = \sqrt{\left\lvert -3 \times tg(x) \times lg(x^4 + y) / e^{-x} + 1 \right\rvert}$
13	$n = \sqrt{e^x + tg(x) + 1} \times \left(lg(y) + cos(x \times y) + \sqrt[3]{x}\right)$
14	$p = \frac{lg(x) - e^{x+y}}{\sqrt{2} + y^2 + \lvert x^3 - ln(y) \rvert}$
15	$q = \sqrt{12 \times x^4 - 3 \times x^3 + 4 \times x^2 - 5 \times x + 6} - lg^2(z)$
16	$r = lg\left\lvert 1 - 2 \times x + 3 \times x^2 - 4 \times x^3 \right\rvert + \sqrt{\lvert x \rvert} / z$
17	$s = \frac{2 \times cos(x - 1/6)}{1/2 + sin^2(y)} - \frac{1}{\lvert x^2 / (y + x^3) \rvert}$
18	$t = \frac{x \times y \times z - y \times \lvert x + \sqrt{z} \rvert}{10^7 + \sqrt[4]{lg(4)}}$
19	$u = \frac{(x + y - z)^3 - (x - y + z)^2 + \sqrt{\lvert x + y + z \rvert}}{log_2(tg(2))}$
20	$w = \frac{(x / y) \times (z + x) \times e^{\lvert x - y \rvert} + ln(1 + e)}{sin^2(y) - (sin(x) \times sin(y))^2}$

ЛАБОРАТОРНАЯ РАБОТА №5

Оператор условного перехода IF

Цель работы:

1) *Изучение структуры и принципа действия оператора условного перехода IF.*
2) *Ознакомление с правилами записи операций отношения и логических выражений.*
3) *Приобретение навыков программирования вычислительных процессов, разветвляющихся в зависимости от выполнения заданного условия.*

Алгоритм называется ***разветвленным***, если он содержит несколько ветвей, отличающихся друг от друга содержанием вычислений. Выход вычислительного процесса на ту или иную ветвь алгоритма определяется исходными данными задачи.

Для разветвления вычислительного процесса используют операторы IF, CASE, GOTO.

Существуют две разновидности условного оператора IF, ***общая форма записи*** которых следующая:

1) IF *условие* THEN *оператор 1*
   ELSE *оператор 2*;
2) IF *условие* THEN *оператор 3;*

где *условие* - выражение булевского типа (логическое выражение);

оператор 1, оператор 2, оператор 3 - операторы языка Pascal, причем на их месте может стоять и составной оператор, т.е. последовательность операторов, заключенных в операторные скобки BEGIN...END.

Суть действия оператора заключается в следующем.

Для первого вида: если *условие* выполняется (значение логического выражения - истина, или TRUE), тогда выполняется *оператор 1*; если *условие* не выполняется (значение логического выражения - ложь, или FALSE), то *оператор 1* не выполняется, а выполняется *оператор 2*, следующий за ELSE. Иными словами, **если** соблюдается некоторое *условие*, то выполняй *оператор 1*, **иначе** (ELSE) выполняй *оператор 2*.

Например,

```
IF X>Y THEN X:=5.1 ELSE Y:=0.0;
```

Пусть из предварительных расчетов известно, что X=10 и Y=8. Логическое выражение X>Y - истинно, поэтому X присваивается новое значение 5.1, а Y останется без изменения.

Для случая, когда X=10 и Y=20, условие X>Y не выполняется т.е. значение логического выражения - ложь; поэтому X остается без изменений и выполняется оператор Y:=0.0.

Для второго вида записи оператор IF работает следующим образом: если *условие* соблюдается (логическое выражение истинно), выполняется *оператор 3*, следующий за THEN, если значение выражения - ложь, тогда выполняется оператор, следующий сразу же за оператором IF. Иными словами, **если** *условие* соблюдается, то выполняй *оператор 3*.

Например,

```
IF X>Y THEN X:=10.0 ; Y:=20.0 ;
```

Пусть X=3 и Y=1, тогда логическое выражение X>Y принимает значение истина, т.е. условие выполняется. Поэтому далее выполняется оператор X:=10.0, а затем и оператор Y:=20.0, следующий за оператором IF.

Для случая, когда X=3 и Y=5 , условие X>Y не выполняется. Поэтому X остается без изменений (X=3), а выполняется оператор, следующий за IF, т.е. Y:=20.

Оператор IF может быть ***вложенным***, т.е. на месте *оператора 1* или *оператора 2* может стоять другой оператор IF.

Например,

```
IF  условие 1  THEN оператор 1
                ELSE
                    IF условие2 THEN оператор 2
                                ELSE  оператор 3;
```

ЗАДАНИЕ.

Разработать программу для вычисления выражения в соответствии с заданным вариантом (см. табл.3).

Ввод исходных данных осуществить с клавиатуры, предусмотрев диалоговый режим работы. Результаты расчета вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета.

Таблица 3

Варианты заданий

Номер варианта	Выражение	Исходные данные
1	$a=\begin{cases}(x+y)^2-\sqrt{x\times y}\ , \text{если}\ \ x\times y>0;\\ (x+y)^2-\sqrt{\lvert x\times y\rvert}\ , \text{если}\ \ x\times y<0;\\ (x+y)^2+1 \qquad , \text{если}\ \ x\times y=0\end{cases}$	x, y
2	$b=\begin{cases}ln(x/y)+(x^2+y)^3 , \text{если}\ x/y>0;\\ ln\lvert x/y\rvert+(x^2+y)^3 , \text{если}\ x/y<0;\\ (x^2+y)^3 \qquad , \text{если}\ x=0;\\ 0 \qquad , \text{если}\ y=0\end{cases}$	x, y
3	$c=\begin{cases}a\times x^3+b\times lg\lvert 2\times x\rvert \quad , \text{если} \quad \sqrt{\text{a - b}}<x;\\ \sqrt{a+sin(2\times x)}-e^{\lvert 3\times x\rvert} , \text{если} \quad \sqrt{\text{a - b}}=x;\\ \dfrac{arctg(5\times x)}{b\times cos(x)+ln(a\times x)} \quad , \text{если} \quad \sqrt{a-b}>0\end{cases}$	a, b, x
4	$d=\begin{cases}ln(x^2)-e^{a\times x+b}+\lg\lvert \text{a - b}\rvert \qquad , \text{если} 2\times a+b<0;\\ tg(a\times x-b^2)-b\times\lvert x^{-7}\rvert \qquad , \text{если}\ 2\times a+b=0;\\ arctg(2\times x-0{,}5)^2+\sqrt{a+b\times x} , \text{если}\ 2\times a+b>0\end{cases}$	a, b, x
5	$f=\begin{cases}e^{sin(x)}+b\times\sqrt{2\times cos(6x-0{,}3)} , \text{если}\ a^2\times x<b^3;\\ (x^2-a)\times sin(x)-2^8 \qquad , \text{если}\ a^2\times x=b^3;\\ e^{-x}+\sqrt{tg(3x+0{,}6)} \qquad , \text{если}\ a^2\times x>b^3\end{cases}$	a, b, x
6	$g=\begin{cases}x^2+y^2+sin(x) , \text{если}\ \ x-y=0;\\ (x-y)^2+cos(x), \text{если}\ \ x-y>0;\\ (y-x)^2+tg(x) , \text{если}\ \ x-y<0\end{cases}$	x, y
7	$h=\begin{cases}(x-y)^3+arctg(x) , \text{если}\ \ x>y;\\ (y-x)^3+arctg(x) , \text{если}\ \ y>x;\\ (x+y)^3+0{,}5 \qquad , \text{если}\ \ x=y\end{cases}$	x, y
8	$j=\begin{cases}i\times\sqrt{a} \qquad , \text{если i - нечетное, a}>0;\\ i/2\times\sqrt{\lvert a\rvert} , \text{если i - четное, a}<0;\\ \sqrt{\lvert i\times a\rvert} \qquad , \text{иначе}\end{cases}$	i, a

Продолжение табл. 3

Номер варианта	Выражение	Исходные данные
9	$l=\begin{cases} e^{\lvert a\rvert-\lvert b\rvert}, \text{если } 0{,}5<\text{a}\times\text{b}<10; \\ \sqrt{\lvert a+b\rvert}, \text{если } 0{,}1<\text{a}\times\text{b}<0{,}5; \\ 2\times\text{x}^2, \text{иначе} \end{cases}$	a, b, x
10	$k=\begin{cases} arctg(x+\lvert y\rvert), \text{если } x<y; \\ arctg(\lvert x\rvert+y), \text{если } \text{x}>y; \\ (x+y)^2, \text{если } x=y \end{cases}$	x, y
11	$m=\begin{cases} sin(5\times k+3\times d\times\lvert k\rvert), \text{если } -1<k<d; \\ sin(5\times k+3\times d\times\lvert k\rvert), \text{если } \text{k}>d; \\ k^3, \text{если } k=d \end{cases}$	k, d
12	$n=\begin{cases} 3\times k^3+3\times p^2, \text{если} k>\lvert p\rvert; \\ \lvert k-p\rvert, \text{если } 3<\text{k}<\lvert\text{p}\rvert; \\ (k-p)^2, \text{если } k=\lvert p\rvert \end{cases}$	k, p
13	$p=\begin{cases} ln(\lvert f\rvert+\lvert q\rvert), \text{если} \lvert\text{f}\times\text{q}\rvert>10; \\ e^{f+q}, \text{если } \lvert\text{f}\times\text{q}\rvert<10; \\ f+q, \text{если} \lvert\text{f}\times\text{q}\rvert=10 \end{cases}$	f, d
14	$r=\begin{cases} a^x-e^x+b^3+\cos(4\times x-0{,}2), \text{если} \lvert\text{a}^2-b^2\rvert<10\times\text{x}; \\ b\times\lvert x^3-a\rvert/(x^3-a)-e^{4x}, \text{если} \lvert\text{a}^2-b^2\rvert=10\times\text{x}; \\ tg(4{,}5\times x)+\dfrac{\lvert x^{-9}\rvert}{sin(0{,}5\times x)}, \text{если} \lvert\text{a}^2-b^2\rvert>10\times\text{x} \end{cases}$	a, b, x
15	$s=\begin{cases} a\times\sqrt{sin(x)+cos^2(x)}+e^{a+b\times x}, \text{если } \sqrt{b\times x^3-65}<a; \\ 1-ln(\sqrt{a\times x^2-b})-\lvert x^6\rvert, \text{если } \sqrt{b\times x^3-65}=a; \\ tg(4{,}5\times x)+\dfrac{\lvert x^{-9}\rvert}{sin(0{,}5\times x)}, \text{если } \sqrt{b\times x^3-65}>a \end{cases}$	a, b, x
16	$t=\begin{cases} ln(x^2)-e^{a\times x+b}+lg\lvert a-b\rvert, \text{если } 2\times\text{a - b}<0; \\ tg(a\times x-b^2)-b\times\lvert x^{-7}\rvert, \text{если } 2\times\text{a - b}=0; \\ arctg(2\times x+0{,}5)+\sqrt{a+b\times x}, \text{если } 2\times\text{a - b}>0 \end{cases}$	a, b, x

ЛАБОРАТОРНАЯ РАБОТА №6

Программирование разветвляющихся алгоритмов

Цель работы:

1) *Закрепление навыков программирования вычислительных процессов, разветвляющихся в зависимости от выполнения условия.*

ЗАДАНИЕ.

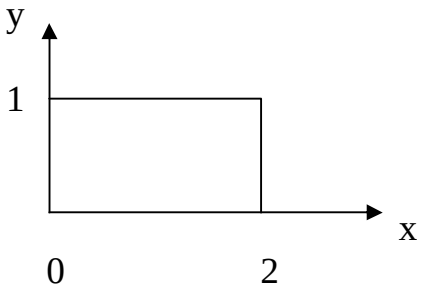
Разработать программу для определения принадлежности точки заданной области в соответствии с заданным вариантом. Если точка принадлежит заданной области, найти значение функции в заданной точке.

Ввод исходных данных осуществить с клавиатуры, предусмотрев диалоговый режим работы. Результаты расчета вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета.

ВАРИАНТЫ ЗАДАНИЯ

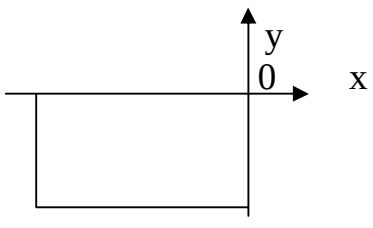
1. Принадлежит ли точка области:

$Y = 2^{-x}\sqrt{x + \sqrt[4]{|x|}}$, при X = 1.741



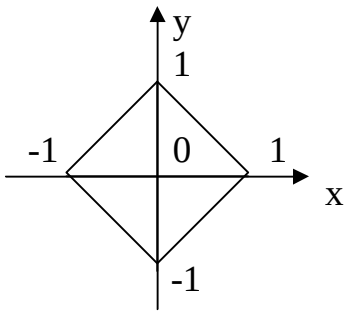
2. Принадлежит ли точка области:

$Y = \sqrt[3]{e^x - \sin x}$, при x = -1.312



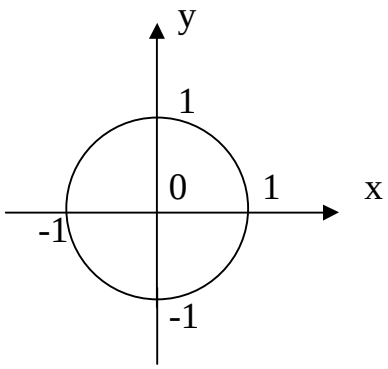
3. Принадлежит ли точка области:

$Y = \sqrt{|x-1| + sin^2 x}$, при x = 0.7409



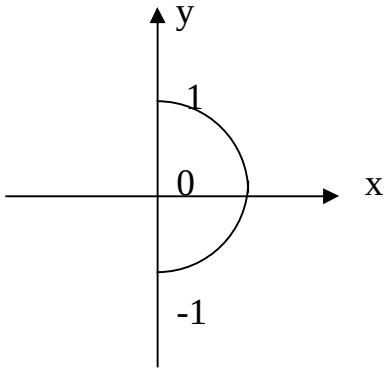
4. Принадлежит ли точка области:

$Y = x\cos x + sin^3 x$ при x = 0.872



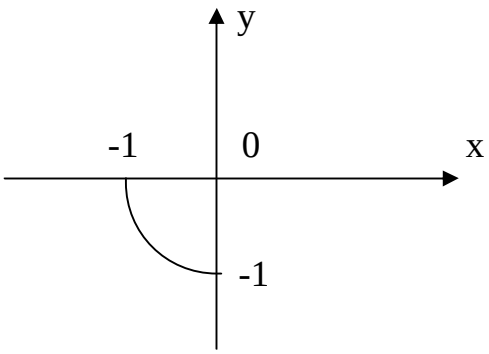
5. Принадлежит ли точка области:

$Y = tgx + |x|$, при x = 0.6312



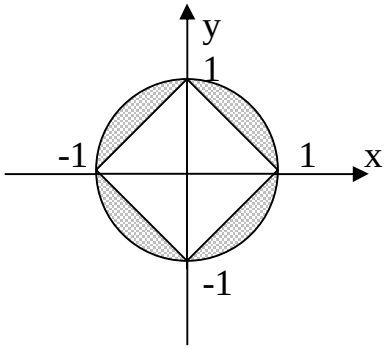
6. Принадлежит ли точка области:

$Y = 1 + \frac{1}{x} + \frac{1}{x^2}$ при x = -0,386



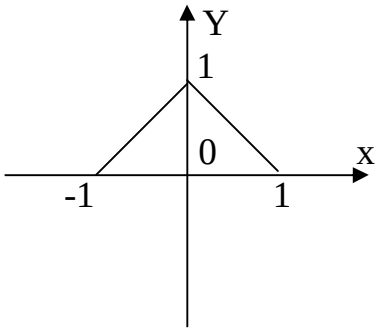
7. Принадлежит ли точка области:

$Y = ch|x+1|$, при x = 0.4589



8. Принадлежит ли точка области:

Y=5arctgx, при x=-0.811



ЛАБОРАТОРНОЯ РАБОТА №7

Оператор выбора CASE

Цель работы:

1) Изучение структуры и принципа действия оператора выбора CASE.

2) Приобретение навыков программирования вычислительных процессов, разветвляющихся в зависимости от выполнения заданного условия.

Оператор выбора CASE является обобщением оператора IF и позволяет сделать выбор из *произвольного числа имеющихся вариантов*. Он состоит из выражения, называемого *селектором*, и списка параметров, каждому из которых предшествует список *констант выбора* (список может состоять из одной константы). Как и в операторе IF, здесь может присутствовать слово ELSE, имеющее тот же смысл.

Общая форма записи:

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

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 относится к одному из целочисленных (находящихся в диапазоне –32768....32767), булевскому, литерному или пользовательскому, типов. Вещественные и строковые типы в качестве селектора запрещены. Список констант выбора состоит из произвольного количества значений и диапазонов, отделенных друг от дрга запятыми. границы диапазона записываются двумя константами через разграничитель ..(две точки). Тип констант в любом случае должен совпадать с типом селектора. Ниже приведены ***примеры*** типичных форм 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;
```

Аналогичным способом можно организовать ввод и вывод данных перечисляемого типа пользователя и обойти соответствующие ограничения языка PASCAL.

ЗАДАНИЕ

Разработать программу для вычисления значения переменной в соответствии с заданным вариантом (см. табл. 4).

Ввод исходных данных осуществить с клавиатуры, предусмотрев диалоговый режим работы. Результаты расчета вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета.

Таблица 4

Варианты заданий

№	Выражение	Исходные данные
1	$L=\begin{cases} e^{\lvert a\rvert-\lvert b\rvert}+e^{\lvert b\rvert-\lvert a\rvert}, если\ m=1 \\ \sqrt{\lvert a+3a*b+b\rvert}, если\ m=2 \\ 2*x^3, иначе \end{cases}$	a, b, x, m
2	$L=\begin{cases} arctg(2x+\lvert y\rvert-\lvert x*y\rvert), если\ m=1; \\ arctg(\lvert x\rvert+2y+x*y), если\ m=2; \\ (x+y)^4, если\ m=3; \end{cases}$	x, y, m
3	$m=\begin{cases} cos(5*k+3*d*\lvert k\rvert-d*k), если\ m=1; \\ sin^2(5*k+3*d*\lvert k\rvert), если\ m=2; \\ k^3, если\ m=3; \end{cases}$	k, d, m
4	$N=\begin{cases} 3*k^p+3*p^k, если\ m=1; \\ \lvert k-2p\rvert, если\ m=2; \\ (3k-p)^2, если\ m=3; \end{cases}$	k, p, m
5	$P=\begin{cases} ln^2(\lvert f\rvert+\lvert q\rvert), если\ m=1; \\ e^{(f+q)/q}, если\ m=2; \\ \lvert f-q\rvert, если\ m=3; \end{cases}$	f, q, m
6	$S(m,x)=\begin{cases} 1-ln(\sqrt{ax^2-b})-\lvert x^2\rvert*b, если\ m=2 \\ x\sqrt{tg(x)+sin^2(x)}+e^{2a+bx}, если\ m=1 \\ 1/tg(4,5x)+\dfrac{\lvert x-9\rvert}{sin^2(0,5x)}, если\ m=3 \end{cases}$	a,x,b,m
7	$T(m,x)=\begin{cases} ln(x^2)-e^{ax+b}+ln\lvert a-b\rvert-tg\lvert b+a^2\rvert, если\ m=1 \\ tg(3ax-b^2+bx)-b\lvert x^{-7}\rvert, если\ m=2 \\ arctg(2x-0,5)+\sqrt{a+bx}+e^x, если\ m=3 \end{cases}$	a,x,b,m
8	$R=\begin{cases} a^{x+3}-e^{x-1}+b^3*e^x+cos(4*x-0.2b), если\ m=1; \\ 2a*tg(4.5*x)+\dfrac{\lvert x^{-9}\rvert}{cos^3(0.5*x)}, если\ m=3; \\ ab*\lvert x^3-a\rvert/(x^3-a)-e^{4*x}, если\ m=2; \end{cases}$	a, b, x, m

ЛАБОРАТОРНАЯ РАБОТА №8

Программирование циклических алгоритмов с заданным количеством шагов. Вычисление суммы ряда

Цель работы:

1) *Изучение структуры и порядка действия оператора цикла с параметром.*
2) *Приобретение навыков в организации циклических вычислительных процессов с фиксированным количеством повторений.*

Алгоритм называется ***циклическим***, если он предусматривает многократное выполнение одних и тех же действий при различных значениях промежуточных данных. Число повторений может быть задано в ***явной*** или ***неявной*** форме. Если число повторений тела цикла заранее известно, то чаще всего применяется ***оператор цикла с параметром***.

Общий вид оператора цикла с параметром:

FOR I:=S1 **TO** S2 **DO** OP;

или

FOR I:=S1 **DOWNTO** S2 **DO** OP;

где FOR...DO - заголовок цикла;

OP - *тело цикла*, т.е. многократно повторяющаяся часть программы; телом цикла может быть простой или составной оператор;

I - *параметр цикла*: переменная, которая изменяется в ходе выполнения оператора цикла при каждом новом выполнении тела цикла;

S1 и S2 - выражения, задающие *начальное и конечное значения* параметра цикла, т.е. параметр цикла изменяется от S1 до S2 с заданным шагом, причем по первому варианту шаг равен, т.е. цикл осуществляется с наращиванием параметра цикла, а по второму варианту (DOWNTO) шаг равен -1, т.е. осуществляется убывание параметра цикла.

Суть действия оператора заключается в следующем. Параметру I присваивается значение S1. С этим значением выполняется тело цикла. Затем параметру цикла присваивается значение I=S1+1 (или S1-1 для DOWNTO), и тело цикла выполняется с новым значением I. Так продолжается до тех пор, пока I не станет больше S2, после чего управление передается оператору, следующему за оператором цикла. Выход из цикла может быть осуществлен также с помощью оператора перехода по метке (GOTO), расположенной вне данного цикла. Параметр цикла I, S1 и S2 должны принадлежать к одному и тому же типу данных (нельзя использовать переменные вещественного типа).

Пример использования оператора цикла с параметром:

```
FOR k:=4 TO 8 DO   Begin  k1:=3*k; WRITELN ('k1=',k1)  End;
```

Здесь I=k, S1=4, S2=8, тело цикла представляет собой составной оператор, заключенный в операторные скобки. Сначала k принимает значение 4. Затем вычисляется значение k1= 3×k = 3×4 = 12. Затем осуществляется вывод

на дисплей записи k1=12. Далее наращиваем k, т.е. k = k+1 = 5. Вычисляется k1=3k=3×5=15; выводится на дисплей k1=15 и т.д., пока k не станет больше 8.

С убыванием параметра цикла:

FOR k:=8 DOWNTO 4 DO BEGIN k1:=3*k; WRITELN ('k1=',k1) END;

Сначала k=8; k1=3×k=24. Выводится на дисплей запись k1=24. Затем k=k-1=8-1=7; k1=3×k=3×7=21. Выводится запись k1=21 и т.д. до k=4.

ЗАДАНИЕ.

Разработать программу для вычисления суммы ряда, состоящего из N элементов. Общий член ряда определяется по рекуррентной формуле в соответствии с заданным вариантом (см. табл.5). Ввод данных осуществить с клавиатуры в диалоге. Результаты расчета вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета.

Таблица 5

Варианты заданий

№варианта	Формула для определения общего члена ряда	Количество элементов N
1	$a_n = (n!)^2 / n^{\sqrt{n}}$	7
2	$a_n = ln(n!) / n^2$	10
3	$a_n = n^{ln(n)} / (ln\, n)^n$	12
4	$a_n = n! / n^{\sqrt{n}}$	5
5	$a_n = e^{-\sqrt[3]{n}}$	15
6	$a_n = n^2 \times e^{-\sqrt{n}}$	20
7	$a_n = (n!) / (3 \times n)^3$	8
8	$a_n = (n!)^3 / e^{\sqrt{n}}$	5
9	$a_n = 10^n / n!$	20
10	$a_n = (n!)^2 / e^{\sqrt{n}}$	10
11	$a_n = ln(n!)^2 / (ln\, n)^n$	7
12	$a_n = 2^n \times (n!) / n^3$	6
13	$a_n = \sqrt{(n!)} \times 3^n / e^{\sqrt{n}}$	15
14	$a_n = e^{\sqrt{n}} / (2 \times n)!$	4
15	$a_n = (n!)^{\sqrt{n}} / e^{\sqrt{n!}}$	7
16	$a_n = 1/2^n + 1/3^{\sqrt{n}}$	5

ЛАБОРАТОРНАЯ РАБОТА №9

Программирование циклических алгоритмов с числом повторений, заданным в неявной форме Вычисление суммы ряда с заданной точностью

Цель работы:

1) *Изучение структуры и принципа действия операторов цикла с предусловием и с постусловием.*
2) *Приобретение навыков в организации циклических вычислительных процессов с числом повторений, заданным неявно.*

Если число повторений тела цикла заранее не известно, то рекомендуется применять ***операторы цикла с предусловием или с постусловием***.

Общая форма записи оператора цикла *с предусловием:*

WHILE *условие* DO *тело цикла*;

где WHILE ... DO ...- заголовок оператора;

условие - выражение булевского типа (логическое);

тело цикла - простой или составной оператор.

Оператор используется, если шаг изменения параметра отличен от 1 или заранее не известно количество повторений тела цикла.

Суть действия оператора заключается в следующем. Выполнение оператора начинается с проверки условия (поэтому его называют оператором цикла с предусловием). Если логическое выражение имеет значение TRUE (истина), то выполняется тело цикла. В теле цикла обязательно должен изменяться один из параметров, оказывающих влияние на значение проверяемого условия, иначе цикл будет повторяться бесконечно. Когда наступает момент, что условие принимает ложное значение FALSE, тогда выполнение цикла прекращается, а управление передается первому после WHILE оператору. Таким образом, действие оператора сводится к следующему: **в то время, пока** соблюдается заданное *условие*, **повторяй** выполнение операторов, запрограммированных *в теле цикла*.

Пример использования оператора цикла *с предусловием*:

```
K:=0;
WHILE K<=10 DO
BEGIN K:=K+2; WRITE(K) END;
```

Предварительно задаем К=0. Затем проверяем условие К<=10 (0<=10, т.е. значение логического выражения - TRUE), тогда К=К+2=0+2=2, и на дисплей выводится цифра 2. Далее проверяем условие К<=10 (2<=10 - TRUE), тогда К=К+2=2+2=4, выводится цифра 4, и т.д. Когда выведется цифра 10, аналогичным образом проверяем условие К<=10 (10<=10 - TRUE),

K=K+2=10+2=12, выводится 12; проверяем K<=10 (12<=10 - FALSE) и цикл завершается. В результате получим последовательность чисел 2,4,6,8,10,12.

Общая форма записи оператора цикла *с постусловием:*

REPEAT *тело цикла*
UNTIL *условие* ;

где REPEAT ... UNTIL - заголовок оператора;
условие - выражение булевского типа (логическое);
тело цикла - некоторые операторы.

Назначение и принцип действия оператора REPEAT аналогичны оператору WHILE, но отличительной особенностью является то, что условие проверяется после выполнения цикла.

Суть действия оператора заключается в следующем. Работа оператора начинается с выполнения тела цикла, и только после этого проверяется значение логического выражения. Если результат булевского выражения, определяемого *условием*, равен FALSE, цикл выполнится еще раз, если TRUE - происходит выход из цикла. Хотя бы один из операторов цикла должен влиять на значение *условия* для предотвращения бесконечного выполнения цикла. Таким образом, действие оператора сводится к следующему: **повторяй** выполнение операторов, запрограммированных в *теле цикла*, **до тех пор**, пока не выполнится заданное *условие*.

В цикле *с предусловием* выход из цикла осуществляется, если условие принимает значение FALSE, из цикла *с постусловием* - если условие принимает значение TRUE. В цикле REPEAT нет необходимости ставить операторные скобки.

Пример использования оператора цикла с постусловием:

```
K:=0;
REPEAT
K:=K+2; WRITE(K)
UNTIL K>=10;
```

Предварительно задаем К=0. Затем вычисляем в цикле К=К+2=0+2=2, и на дисплей выводится цифра 2. Далее проверяем условие К>=10 (2>=10 - FALSE), цикл продолжается. Вычисляем К=К+2=2+2=4, выводится цифра 4, и т.д. Когда выведется цифра 10, аналогичным образом проверяем условие K>=10 (10>=10 - TRUE), и цикл завершается. В результате получим последовательность чисел 2,4,6,8,10.

ЗАДАНИЕ

Разработать программу для вычисления суммы ряда, каждый элемент которого определяется по рекуррентной формуле в соответствии с заданным вариантом (см. табл.6).

Точность вычислений ε=0.00001. Считать, что требуемая точность достигнута, если очередное слагаемое оказалось по модулю меньше, чем ε.

Ввод исходных данных осуществить с клавиатуры, предусмотрев диалоговый режим работы. Результаты расчета вывести на дисплей, снабдив их соответствующими комментариями. Вывести все учитываемые элементы ряда и их количество. Предусмотреть форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета.

Таблица 6

Варианты заданий

№ варианта	Формула для определения общего члена ряда
1	$a_n = (-1)^{n-1} / n^n$
2	$a_n = 1/2^n + 1/3^n$
3	$a_n = (2n-1)/2^n$
4	$a_n = 1/((3n-2)\times(3n+1))$
5	$a_n = 10^n / n!$
6	$a_n = (n!)/(2n)!$
7	$a_n = (n!)/n^n$
8	$a_n = 2^n \times n!/(n^n)!$
9	$a_n = 3^n \times n!/(3n)!$
10	$a_n = n!/(3n^n)$
11	$a_n = (n!)^2/(2^{n^2})!$
12	$a_n = (-1)^n/(n+1)!$
13	$a_n = (-1)^n \times 3^{2n}/(2n)!$
14	$a_n = e^{\sqrt{n}}/(2n)!$
15	$a_n = (-1)^k \times \left(\frac{5}{(n+1)\times(n+2)}\right)^n$
16	$a_n = (-1)^k \times \left(\frac{2}{(n+1)\times(n+2)\times(n+2)}\right)^{2n}$

ЛАБОРАТОРНАЯ РАБОТА №10

Ввод-вывод данных через внешний файл

Цель работы:

1) *Ознакомление со стандартными процедурами для работы с внешними файлами.*
2) *Изучение порядка чтения и записи данных через внешний файл.*
3) *Приобретение навыков оформление отчетного файла по результатам выполнения расчетов.*

Файл – это любой набор элементов одного и того же типа. Файлы могут быть *последовательного* и *прямого доступа*. В файлах последовательного доступа каждый элемент может быть прочитан только после перебора всех предыдущих. В файлах прямого доступа можно обратиться к заданному элементу непосредственно, указав его номер в файле. По отношению к программе файлы могут быть *внешними и внутренними*. Внутренние файлы - те, которые существуют только во время работы программы и исчезают после выполнения программы (например, стандартные файлы ввода вывода).

Внешние файлы хранятся на внешних носителях (например, на жестком диске) и могут существовать отдельно от программы. Они должны быть описаны в разделе описаний программы.

TYPE *имя типа* = FILE OF *базовый тип;*
VAR *имя файла*: *имя типа*;

или

VAR *имя файла*: FILE OF *базовый тип*.

Здесь - *базовый тип* – тип данных, входящих в файл.

Например,

```
TYPE FD = FILE OF REAL;
VAR F1:FD;
```

или

```
F1:  FILE OF REAL;
```

Для работы с внешними файлами предусмотрен ряд стандартных процедур.

ASSIGN - процедура присоединения внешнего файла. Она связывает файловую переменную с именем внешнего файла и его адресом. Эта команда должна быть записана в программе в разделе операторов до работы с внешним файлом:

ASSIGN(*имя*, '*адрес*');

где *имя* - имя файловой переменной;

адрес - имя накопителя, где хранится файл;(указывается в апострофах). Например,

```
ASSIGN(F1, 'D:\USERS\AA\LAB00.DAT');
ASSIGN(F1, 'LAB10.DAT');
```

RESET(*имя*) - открыть файл для чтения (подготовить файл для считывания из него данных). Записывается перед оператором чтения данных. Например, RESET(F1).

REWRITE(*имя2*) - открыть файл для записи (подготовить файл к приему некоторой информации, которая получается в результате выполнения программы и должна сохраниться на длительный срок). Например, REWRITE(F2). Обращение к процедуре осуществляется перед первой командой записи в файл.

READ(*имя, пер1, пер2*) - считывание списка переменных из внешнего файла. Например, READ(F1,a,b,c) - считать переменные a,b,c из внешнего файла F1. Если F1 отсутствует (нет имени), автоматически подключается ввод со стандартного файла (или устройства), т.е. клавиатуры.

WRITE(*имя2, пер1, пер2*) - запись списка переменных во внешний файл. Например, WRITE(F2,X1,X2,X3) - записать значения переменных X1,X2,X3 во внешний файл F2.

CLOSE(*имя*) - закрыть файл. Обычно ставится в конце программы или после последнего READ или WRITE.

Чтение файла - ввод данных из внешнего файла в оперативную память ЭВМ для выполнения программы. Такой файл называется входным. Для чтения данных из внешнего файла необходимо:

1) описать файловую переменную;
2) присоединить файл;
3) открыть файл для чтения;
4) считать данные из файла;
5) закрыть файл.

Например:

```
ASSIGN(F, 'lab3dat.pas')
RESET(F1);
READ(F1,A,B);
CLOSE(F1).
```

Запись в файл – это вывод данных из оперативной памяти ЭВМ на внешний носитель в ходе выполнения программы. Такой файл называется выходным. Для записи данных во внешний файл необходимо:

1) описать файловую переменную;
2) присоединить файл;
3) открыть файл для записи;
4) записать данные в файл;
5) закрыть файл.

Например:

```
ASSIGN(F2, 'REZ10.PAS');
REWRITE(F2);
WRITE(F2,c,d);
CLOSE(F2).
```

Для оформления технических отчетов по результатам выполнения расчетных задач удобно пользоваться ***текстовыми файлами***. Текстовый файл состоит из обычных символов (букв, цифр). Файл разбивается на строки по записям, как в текстовом документе. Текстовый файл описывается в разделе описаний программы:

VAR *имя файловой переменной*: TEXT;

например, VAR F1:TEXT.

Для работы с текстовым файлом можно использовать дополнительные стандартные процедуры:

WRITELN(*имя*)- конец текущей строки при записи, переход на новую строку;

READLN(*имя*)- переход к началу следующей строки при вводе;

OPPEND(*имя*)- открытие уже существующего текстового файла для добавления данных в конец файла.

Все файлы, кроме текстовых (например, of real) представляют собой набор чисел, символьных или текстовых записей они не воспринимают. С текстовым же файлом можно работать с различными типами данных по аналогии выводом на дисплей, или "ручным" оформлением документов.

ЗАДАНИЕ

Модифицировать выполненные предыдущие лабораторные работы, сохранив номер варианта, изменив организацию ввода-вывода данных. В одной из лабораторных работ (по указанию преподавателя) ввод данных осуществить из внешнего файла. Вывод результатов работы программы во всех лабораторных работах осуществить во внешний файл результатов, параллельно организовав печать основных данных, полученных в ходе выполнения программы, на дисплей. Внешний файл результатов оформить в виде технического отчета. Он должен содержать номер и название лабораторной работы, исходные данные и результаты расчета, снабженные соответствующими комментариями, а также инициалы исполнителей. Текст и числовые значения должны быть отформатированы.

ЛАБОРАТОРНАЯ РАБОТА №11

Построение таблиц значений функций

Цель работы:

1) Закрепление навыков использовании оператора цикла с параметром.
2) Знакомство с методами оптимизации программ.
3) Приобретение навыков оформления результатов инженерных расчетов в виде таблиц.

В ходе выполнения инженерных расчетов во многих случаях результаты должны быть представлены в виде множества значений искомых величин в зависимости от различных значений аргумента (например, потери давления в трубопроводах в зависимости от скорости течения жидкости или от диаметра трубопровода). Для удобства оценки полученных характеристик и наглядности представленной информации в технических отчетах целесообразно результаты расчетов представлять в табличной форме. Кроме того, представленные в табличной форме данные могут быть скопированы в один из графических пакетов (EXCEL, GRAPHER) для построения графиков и диаграмм.

Общий алгоритм получения таблицы заключается в многократном вычислении и выводе на печать (дисплей) значений функции при изменяющемся с помощью оператора цикла аргументе X от некоторого начального до максимального его значения с заданным шагом.

Для задания значений X и соответствующих значений функции следует использовать простые переменные. Значение шага H должно вычисляться один раз. При изменении значения аргумента X рекомендуется использовать оператор присваивания X:=X+H, а не оператор с использованием операции умножения X:=A+I∗H, что существенно сокращает время выполнения программы.

Для вычисления значений функции в данной лабораторной работе рекомендуется использовать оператор цикла с параметром. В «шапку» таблицы заносится информация о наименовании представленной в соответствующем столбце переменной и единицах ее измерения. Вывод «шапки» должен быть организован перед обращением к оператору цикла.

ЗАДАНИЕ

Составить программу вычисления значения функции F(X) на отрезке [A,B] в узловых точках X_i=A+iH, где H=(B-A)/M, M- заданное целое число, определяющее количество расчетных точек на заданном интервале.

Варианты заданий представлены в табл.7.

Ввод исходных данных осуществить с клавиатуры, предусмотрев диалоговый режим. Организовать печать “шапки” и границ таблицы результатов. Исходные данные и результаты расчета вывести во внешний файл, снабдив их соответствующими комментариями. Организовать параллельный вывод основных результатов на дисплей. Обеспечить форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета.

Таблица 7

Варианты заданий

№ варианта	Функция F(X)	Параметры		
		A	B	M
1	x-sin(x)	0	π/2	10
2	sin(x)	π/4	π/2	15
3	cos(x)	π/3	2π/3	20
4	tg(x)	0	π/4	10
5	ctg(x)	π/4	π/2	15
6	arcsin(x)	0	1	20
7	arccos(x)	0.5	1	10
8	arctg(x)	2	7	15
9	sin(x)-cos(x)	0	π/2	20
10	x+sin(x)	0	3π	10
11	sin(1/x)	π/8	2/π	15
12	cos(1/x)	π/4	4/π	20
13	$\sin(x^2)$	π/6	2π/3	10
14	$\cos(x^2)$	π/3	3π/2	15
15	sin(x)+tg(x)	0	π/4	20
16	cos(x)+ ctg(x)	π/4	π/2	10
17	tg(x/2)	0	2π/3	15
18	tg(x/2)+ cos(x)	π/2	π	20
19	ctg(x/3)+ sin(x)	π/4	π/2	10
20	sin(x/4)/2	π/2	π	15

ЛАБОРАТОРНАЯ РАБОТА №12

Ввод-вывод массивов

Цель работы:

1) Изучение данных, относящихся к типу массив.

2) Ознакомление с порядком описания массивов в программах.

3) Приобретение навыков ввода и вывода элементов массива.

Массивом называется упорядоченная совокупность конечного числа данных одного типа. Массив может быть *одномерным* и *многомерным*.

Например, одномерным массивом является последовательность чисел

а) 3 4 5 8 6 9 ,

а двумерным массивом является матрица

б) 3 4 6 8 10
12 6 5 4 3 .

При обозначении массивов все его элементы имеют *одинаковое имя*, но каждый элемент массива имеет *свой индекс*, определяющий место элемента массива в упорядоченной последовательности данных (чисел).

Имя массива составляется так же как и имя переменной (например, из алфавитно-цифровых знаков). Массиву из последовательности чисел а) можно присвоить имя MAS1, а массиву б) - имя MAS2.

Для массивов с постоянным числом элементов размерность массива задается при его описании в начале программы и в дальнейшем не изменяется. Для того чтобы определить *место* элемента в массиве, к имени массива дописываются *индексы в квадратных скобках*.

Массив описывается в разделе описаний программы. Сделать это можно *двумя способами*:

```
1) TYPE имя типа=ARRAY[t1,t2,...,tn] OF type1;
       VAR имя массива: имя типа;
2) VAR имя массива: ARRAY[t1,t2,...,tn] OF type1;
```

где t1, t2, tn - тип *индексов* массива;

type1 -тип *элементов*, из которых состоит массив.

Количество tn определяет *размерность* массива. Количество измерений массива практически не ограничено. Индексы могут относиться, например, к символьному типу char, логическому типу boolean , ограниченному типу 1..10. Нельзя использовать переменные типа REAL и INTEGER, т.к. в этом случае размерность массива не определена.

Значение индекса может быть любым данным скалярного типа, кроме REAL.

Для вычислительных процессов наиболее часто употребляется ограниченные типы индексов массива и целые значения индексов.

Конкретные значения индексов определяют *место элемента в массиве*. Например, массив MAS2 должен быть описан как MAS2[1..2,1..5], т.е. n=2 (двумерный), в котором тип индексов t1 и n2 является ограниченным, а значения индексов - целые числа от 1 до 2 и от 1 до 5 по соответствующим измерениям массива.

Для того, чтобы указать произвольный элемент массива, необходимо записать имя массива и индексы элемента MAS2[I,J], причем I, J должны быть описаны как данные целого типа.

Тип *элементов* массива type1 может быть любым простым или сложным, например, REAL, INTEGER, BOOLEAN и др.

Опишем выбранные массивы в соответствии с указанными двумя правилами. По *первому способу*:

```
TYPE C=ARRAY[1..6] OF INTEGER;
       D=ARRAY[1..2,1..5] OF INTEGER;
VAR MAS1:C;
      MAS2:D;
      MAS3,MAS4,MAS5:D;
```

Эта запись означает следующее. В программе создается некоторый новый тип данных C, представляющий собой одномерный массив, состоящий из 6 элементов целого типа. Все переменные, которые будут относиться к типу C, представляют собой массив указанной структуры. В программе создается также некоторый новый тип данных D, представляющий собой двумерный массив размерностью 2x5, состоящий из 10 элементов целого типа, т.е. все переменные, которые относятся к типу D, будут двумерным массивом указанной структуры.

Далее следует запись, указывающая, что переменная MAS1 относится к типу C, а переменная MAS2 и дополнительно вводимые в программу переменные MAS3, MAS4, MAS5 – к типу D.

В соответствии со *второй* формой массивы опишутся следующим образом:

```
VAR MAS1:ARRAY[1..6] OF INTEGER;
      MAS2:ARRAY[1..2,1..5] OF INTEGER;
```

Ввод-вывод массивов осуществляется с помощью *операторов цикла*. Для n-мерного массива используются n раз вложенные циклы. Причем сначала выполняется внутренний цикл, а затем все последующие внешние цик-

лы. Например, ввод элементов трехмерного массива MAS10 можно осуществить с помощью оператора

```
FOR I:=1 TO Imax DO
    FOR J:=1 TO Jmax DO
        FOR K:=1 TO Kmax  DO
        READ(MAS10[I, J, K]);
```

где Imax, Jmax, Kmax - данные, определяющие размер массива по каждому из измерений.

Массивы могут также *вводиться как типизированные константы*:

```
TYPE имя типа=ARRAY[t1, t2,..., tn] OF type1;
CONST имя константы: имя типа=(список констант);
```

Список констант заключается в круглые скобки, константы отделяются запятой. Для многомерных массивов константы каждой строки заключаются в круглые скобки и отделяются запятой. Например,

```
TYPE VEC=ARRAY[1..2, 1..5] OF INTEGER;
    CONST A:VEC=((3,6,7,1,4),(10,2,6,4,5));
```

ЗАДАНИЕ.

Разработать программу для ввода и вывода на печать элементов одномерного и двумерного массивов. Варианты исходных данных представлены в табл.8.

Ввод исходных данных с клавиатуры осуществить в диалоговом режиме. Результаты выполнения программы вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть форматный вывод результатов, очистку экрана и задержку выполнения программы в конце расчета. Элементы массива выводить с указанием имени массива и соответствующих индексов, например, MAS[2,4]=25.

Таблица 8

Варианты заданий

Номер варианта	Одномерный массив				Двумерный массив				
	К-во элементов	Тип элементов	Способ ввода	Форма вывода	К-во строк	К-во столбцов	Тип элемен-тов	Способ ввода	Форма вывода
1	5	целый	типизированная константа	строка	2	6	вещественный	цикл	матрица
2	7	вещественный	цикл	столбец	3	4	целый	типизированная константа	матрица
3	8	вещественный	типизированная константа	строка	4	5	целый	цикл	матрица
4	4	целый	цикл	столбец	2	6	вещественный	типизированная константа	матрица
5	3	вещественный	типизированная константа	строка	3	5	целый	цикл	матрица
6	6	целый	цикл	строка	4	3	вещественный	типизированная константа	матрица
7	5	целый	типизированная константа	столбец	5	4	вещественный	цикл	матрица
8	4	вещественный	цикл	столбец	4	3	целый	типизированная константа	матрица
9	6	целый	типизированная константа	столбец	3	6	вещественный	цикл	матрица
10	7	целый	цикл	строка	2	6	вещественный	типизированная константа	матрица
11	3	вещественный	типизированная константа	строка	3	5	целый	цикл	матрица
12	6	вещественный	цикл	столбец	4	3	целый	типизированная константа	матрица
13	5	вещественный	типизированная константа	строка	5	3	целый	цикл	матрица
14	4	целый	цикл	строка	6	2	вещественный	типизированная константа	матрица
15	5	вещественный	типизированная константа	столбец	5	4	целый	цикл	матрица
16	7	целый	цикл	строка	4	3	вещественный	типизированная константа	матрица
17	4	целый	типизированная константа	столбец	3	6	вещественный	цикл	матрица
18	5	целый	цикл	строка	5	4	вещественный	типизированная константа	матрица
19	3	целый	типизированная константа	столбец	4	5	вещественный	цикл	матрица
20	6	вещественный	цикл	строка	6	2	целый	типизированная константа	матрица

ЛАБОРАТОРНАЯ РАБОТА №13

Преобразование одномерных массивов

Цель работы:

1) *Изучение алгоритма нахождения максимального (минимального) элемента ряда и его номера.*
2) *Ознакомление с алгоритмом перестановки данных в числовом ряду.*
3) *Приобретение навыков программирования циклических процессов с использованием массивов данных.*

Выполнение данной работы основано на знании правил описания массивов, порядка их ввода-вывода, а также сути действия и формы записи оператора цикла с параметром.

Описать массив в Паскаль-программе можно *двумя способами*:

1) TYPE имя типа=ARRAY[t1,t2,...,tn] OF type1;
   VAR имя массива: имя типа;
2) VAR имя массива: ARRAY[t1,t2,...,tn] OF type1;

где t1, t2, tn - тип *индексов* массива; type1 -тип *элементов* массива.

В вычислительных процессах удобно использовать ограниченные *типы индексов* массива и целые *значения индексов*. Значения индексов определяют место элемента в массиве. Для указания некоторого элемента массива необходимо записать имя массива и индексы элемента.

Ввод-вывод массивов осуществляется с помощью *операторов цикла*. Для n-мерного массива используются n раз вложенные циклы.

Общий вид ***оператора цикла*** с параметром:

FOR I:=S1 **TO** S2 **DO** OP; или **FOR I:**=S1 **DOWNTO** S2 **DO** OP;

где OP - *тело цикла*; I - *параметр цикла*; S1 и S2 - выражения, задающие *начальное и конечное значения* параметра цикла.

В лабораторной работе требуется найти максимальный или минимальный элемент массива и поменять его местами с указанным в соответствии с вариантом задания.

Общий алгоритм решения задачи заключается в следующем.

1) В разделе описаний описываются все типы данных, в том числе массивы.
2) С помощью оператора цикла вводятся элементы массива.
3) Далее необходимо организовать вывод исходного массива.
4) Затем *определяется максимальный (минимальный) элемент массива и его номер* (индекс). Это осуществляется с помощью условного оператора IF. Например, для нахождения минимального элемента сначала предполагается, что таковым является первый. В цикле с минимальным поочередно сравниваются все элементы массива. Если в массиве находится такой, что будет меньше, чем предполагаемый минимальный, то этот элемент массива становится минимальным, причем необходимо запомнить номер этого элемента, присвоив соответствующей переменной текущее значение пара-

метра цикла. Если в дальнейшем появится еще меньший элемент, произойдет переприсваивание минимального значения и номера элемента.

5) После выхода из цикла осуществляется *замена элементов*. Для этого рекомендуется воспользоваться дополнительной переменной (буфером), куда следует записать значение одного из заменяемых элементов массива.
6) В конце программы следует вывести преобразованный массив (после перестановки элементов).

ЗАДАНИЕ

Разработать программу для замены местами элементов одномерного массива. Варианты исходных данных представлены в табл.9.

Ввод исходных данных с клавиатуры осуществить в диалоговом режиме. Результаты выполнения программы вывести на дисплей, снабдив их соответствующими комментариями. Предусмотреть форматный вывод значений элементов массива, очистку экрана и задержку выполнения программы в конце расчета. Элементы массива выводить с указанием имени массива и соответствующих индексов. Вывести исходный и преобразованный массивы.

Таблица 9

Варианты заданий

Номер варианта	Количество элементов	Тип элементов	Искомый элемент массива	Элемент массива, который необходимо поменять местами с искомым
1	8	целый	минимальный	первый
2	6	целый	максимальный	первый
3	7	целый	минимальный	последний
4	9	целый	максимальный	последний
5	5	вещественный	минимальный	первый
6	7	вещественный	максимальный	первый
7	8	вещественный	минимальный	последний
8	6	вещественный	максимальный	последний
9	9	целый	минимальный	второй
10	6	целый	максимальный	второй
11	8	целый	минимальный	предпоследний
12	7	целый	максимальный	предпоследний
13	7	вещественный	минимальный	второй
14	8	вещественный	максимальный	второй
15	9	вещественный	минимальный	предпоследний
16	8	вещественный	максимальный	предпоследний
17	7	целый	минимальный	третий
18	6	целый	максимальный	третий
19	5	вещественный	минимальный	третий
20	7	вещественный	максимальный	третий

ЛАБОРАТОРНАЯ РАБОТА №14

Упорядочивание одномерных массивов

Цель работы:

1) *Изучение алгоритма расстановки элементов ряда в заданном порядке.*
2) *Приобретение навыков программирования вложенных циклических процессов для преобразования массивов данных.*

Основой для решения поставленной задачи может служить лабораторная работа «Преобразование одномерных массивов», в которой проводился обмен местами одного из элементов массива с экстремальным элементом. В данной работе алгоритм усложняется введением дополнительного вложенного цикла с изменяющимся начальным (или конечным) значением параметра цикла, причем указанное значение параметра вложенного цикла задается значением параметра внешнего цикла. Для выполнение данной работы необходимо знать правила описания массивов, порядок их ввода-вывода, а также суть действия оператора цикла с параметром.

Общий алгоритм решения задачи заключается в следующем.

1) В разделе описаний описываются все данные, используемые в программе, в том числе массивы.
2) Вводятся элементы исходного массива.
3) Далее необходимо организовать вывод исходного массива (во внешний файл или на дисплей).
4) Определяется максимальный (минимальный) элемент массива и его номер (индекс).
5) После выхода из цикла, который будет вложенным, осуществляется *замена элементов* (например, сначала максимального с последним, затем, на очередном шаге внешнего цикла – максимального из оставшихся с предпоследним и т.д.).
6) Далее необходимо повторить пункты 4...6 путем организации дополнительного (внешнего) цикла, причем во вложенном цикле (п.4) начальное (или конечное) значение параметра цикла будет изменяться в соответствии со значением параметра внешнего цикла.
7) В конце программы следует вывести преобразованный массив.

ЗАДАНИЕ

Разработать программу для упорядочивания элементов одномерного массива. Например, упорядочить массив по неубыванию означает расположить его элементы так, чтобы каждый последующий элемент был больше или равен предыдущему. Варианты исходных данных представлены в табл.10.

Ввод исходных данных с клавиатуры осуществить в диалоговом режиме. Результаты выполнения программы вывести на дисплей и (или) во внешний файл, снабдив их соответствующими комментариями. Предусмотреть форматный вывод значений элементов массива, очистку экрана и задержку выполнения программы в конце расчета. Элементы массива выводить с указанием имени массива и соответствующих индексов. Вывести исходный и преобразованный массивы.

Таблица 10

Варианты заданий

Номер варианта	Количество элементов	Тип элементов	Порядок расположения элементов в преобразованном массиве
1	7	целый	по неубыванию
2	5	целый	по невозрастанию
3	8	целый	по убыванию
4	9	целый	по возрастанию
5	4	вещественный	по неубыванию
6	6	вещественный	по невозрастанию
7	7	вещественный	по убыванию
8	6	вещественный	по возрастанию
9	9	целый	по неубыванию
10	8	целый	по невозрастанию
11	7	целый	по убыванию
12	6	целый	по возрастанию
13	5	вещественный	по неубыванию
14	4	вещественный	по невозрастанию
15	5	вещественный	по убыванию
16	6	вещественный	по возрастанию

ЛАБОРАТОРНАЯ РАБОТА №15

Программирование с использованием подпрограмм типа функция

Цель работы:

1) Изучение структуры подпрограммы типа функция.

2) Изучение механизма передачи данных в подпрограммах-функциях.

3) Получение навыков программирования с использованием функций.

Подпрограммой называют часть программы, которая выполняет однотипные действия, причем эти действия могут выполняться над различным набором данных. Подпрограмму оформляют в виде специальной структурной единицы.

Использование подпрограмм *позволяет:*

1) уменьшить объем используемой памяти ЭВМ;

2) сделать программу более компактной;

3) осуществить структурный подход к программированию;

4) решать сложные задачи путем подключения к разработке программы нескольких программистов;

5) экономить время на разработке уже созданных наиболее распространенных алгоритмов обработки данных.

В Паскале используют два вида подпрограмм: ***процедуры и функции***. Структура подпрограммы-функции аналогична структуре головной программы и включает разделы: заголовок, раздел описаний и раздел операторов.

Текст подпрограммы-функции в основной программе располагается в разделе описаний, перед разделом операторов, например:

```
PROGRAM LAB;
LABEL...;
VAR. ...;
   {текст подпрограммы-функции}
BEGIN ... END.
```

Подпрограмма-функция может быть записана в *отдельный файл*. Для того, чтобы включить эту подпрограмму в текст головной программы, используют так называемые директивы компилятора (дополнительные указания):

{ $I *имя и маршрут доступа к файлу***}**.

Эта директива записывается перед разделом операторов. Если файл хранится в текущем каталоге, маршрут можно не указывать.

Пример.

Пусть некоторая подпрограмма оформлена в виде отдельного файла PP.PAS:

```
PROGRAM LAB;
VAR...;
```

{$I PP.PAS}-т.е. включить в текст программы LAB текст файла PP.PAS.
Begin.....end.

Если подпрограмма-функция оформляется в виде отдельного файла, в конце ее после end ставят точку с запятой, а не точку, как это делается в головной программе.

Структура подпрограммы-функции имеет следующий вид:

FUNCTION *имя*(*формальные параметры*)**:** *тип результата*;
раздел описаний;
BEGIN
раздел операторов;
END;

Здесь *имя* - название подпрограммы-функции.

Раздел описаний аналогичен разделу описаний головной программы, в нем могут присутствовать и другие подпрограммы.

Формальные параметры - список переменных с указанием их типа. Переменные одного типа разделяются запятой, списки переменных разных типов - точкой с запятой. Те переменные, которые описываются как формальные параметры (в скобках), в разделе описания функции не описываются.

Формальные параметры - это те переменные, с использованием которых написана универсальная последовательность действий (для всех пользователей). Параметры могут быть и фактическими. *Фактические параметры* - это те данные, с которыми работает программист в конкретной программе. Формальные и фактические параметры могут совпадать.

Пример *заголовка* подпрограммы-функции:

FUNCTION F(X,Y:REAL; N:INTEGER): REAL;

Результат выполнения функции - есть *имя* функции, т.е. результат присваивается идентификатору, обозначающему название подпрограммы-функции. Результатом выполнения функции может быть только одно значение, совпадающее с именем этой функции, поэтому в заголовке указывается тип имени, т.е. *тип результата*. В разделе операторов функции должен обязательно присутствовать оператор присваивания какого-либо значения переменной, обозначающей имя функции.

Вызов подпрограммы-функции осуществляется непосредственно в арифметическом выражении указанием ее имени и перечислением фактических параметров:

имя(*фактические параметры***);**

Список фактических параметров подпрограммы-функции должен соответствовать списку формальных параметров по их количеству, типу и последовательности перечисления.

Пример.

Вычислить значение выражения:

$$y = \frac{sin(a)+cos(a)}{ln(a)} + \frac{sin(a+b)+cos(a+b)}{ln(a+b)} + \frac{sin(a\times b)+cos(a\times b)}{ln(a\times b)}$$

Однотипное преобразование данных $f(x) = \frac{sin(x)+cos(x)}{ln(x)}$ можно оформить в виде подпрограммы-функции.

```
FUNCTION F(x:real):real;
begin
F:=(sin(x)+cos(x))/ln(x)
end;
```

Если эту подпрограмму записать в отдельный файл, например F.PAS, то программа для вычисления заданного выражения будет иметь вид:

```
 PROGRAM V;
VAR a, b, y : real;
 {$i F.PAS}
 BEGIN
    READLN(a,b);
    y:=F(a)+FUN(a+b)+FUN(a*b);
    WRITELN (y);
end.
```

При разработке программ с использованием функций важно рационально распределять используемые в них переменные. *Глобальными* называются те переменные, которые описаны в головной программе. *Локальными* называются переменные, которые описываются в подпрограммах. Локальные переменные действуют только в той программной единице (подпрограмме), где они описаны. В различных подпрограммах могут использоваться локальные переменные с одинаковым именем. Если переменная описана как глобальная, к ней можно обратиться из любого места программы. Рекомендуется максимально использовать локальные переменные, ограничивая число глобальных.

ЗАДАНИЕ

Разработать программу, которая вычисляет сумму значений заданной в соответствии с вариантом (см. табл. 11) функции F(x,y) . Вычисление F(x,y) оформить в виде функции.

Ввод исходных данных осуществить с клавиатуры в диалоговом режиме. Результаты выполнения программы вывести во внешний файл, снабдив их соответствующими комментариями и предусмотрев форматный вывод числовых значений.

Таблица 11

Варианты заданий

Номер варианта	Выражение для вычисления суммы	Функция	Исходные данные
1	$z= f(a, b)+ f(a^2, b^2)+ +f(a^2-1,b)+ f(a-b, b)+ +f(a^2+b^2, b^2-1)$	$f(u, t) = \begin{cases} u^2 + t^2, \text{если } u > 0 \text{ и } t > 0; \\ u + t^2, \text{если } u \le 0 \text{ и } t \le 0; \\ u - t, \text{если } u > 0 \text{ и } t \le 0; \\ u + t, \text{если } u \le 0 \text{ и } t > 0. \end{cases}$	a, b
2	$z = f(\sin\alpha,a)+ f(\cos\alpha,a)+ +f(\sin\alpha,a-1)+ +f(\sin\alpha- \cos\alpha,a^2-1)+ +f(\sin^2 \alpha-1, \cos\alpha+a)$	$f(u, t) = \begin{cases} u + sin(t), \text{если } u > 0 ; \\ u + t, \text{если } u \le 0. \end{cases}$	a, α
3	$z =f(\sqrt{\lvert x\rvert}, y)+ f(a, b)+ +f(\sqrt{\lvert x\rvert}+1, -y)+ +f(\lvert x\rvert-\lvert y\rvert, x)+ + f(x+y, a+b)$	$f(u, t) = \begin{cases} u + 2t, \text{если } u \ge 0; \\ u + t, \text{если } u \le -1; \\ u^2 - 2t + 1, \text{если } -1 < u < 0. \end{cases}$	x, y, a, b
4	$z= f(\sin(x)+\cos(x),x+y)+ +f(\sin(x),\cos(y))+ + f(x-y,x)+ f(\sin^2(x)-2,a)+ +f(a+3,b+1)$	$f(u, t) = \begin{cases} u + t, \text{если } u > 1; \\ u - t, \text{если } 0 \le u \le 1; \\ t - u, \text{если } u < 0. \end{cases}$	x, y, a, b
5	$z= f(a,b)+ f(a^2,b^2)+ +f(a^2-1,b)+ f(a-b,b)+ +f(a^2+b^2,b^2-1)$	$f(u, t) = \begin{cases} u + t, \text{если } u > 1; \\ u - t, \text{если } 0 \le u \le 1; \\ t - u, \text{если } u < 0. \end{cases}$	a, b
6	$z =f(\sin\alpha,a)+ f(\cos\alpha,a)+ +f(\sin\alpha,a-1)+ +f(\sin\alpha- \cos\alpha,a^2-1)+ +f(\sin^2 \alpha-1, \cos\alpha+a)$	$f(u, t) = \begin{cases} u + 2t, \text{если } u \ge 0; \\ u + t, \text{если } u \le -1; \\ u^2 - 2t + 1, \text{если } -1 < u < 0. \end{cases}$	a, α
7	$z =f(\sqrt{\lvert x\rvert}, y)+ f(a, b)+ +f(\sqrt{\lvert x\rvert}+1, -y)+ +f(\lvert x\rvert-\lvert y\rvert, x)+ +f(x+y, a+b)$	$f(u, t) = \begin{cases} u + sin(t), \text{если } u > 0 ; \\ u + t, \text{если } u \le 0. \end{cases}$	x, y, a, b
8/	$z=f(\sin(x)+\cos(x),x+y)+ +f(\sin(x),\cos(y))+ +f(x-y,x)+ f(\sin^2(x)-2,a)+ +f(a+3,b+1)$	$f(u, t) = \begin{cases} u^2 + t^2, \text{если } u > 0 \text{ и } t > 0; \\ u + t^2, \text{если } u \le 0 \text{ и } t \le 0; \\ u - t, \text{если } u > 0 \text{ и } t \le 0; \\ u + t, \text{если } u \le 0 \text{ и } t > 0. \end{cases}$	x, y, a, b

Продолжение табл. 11

Номер варианта	Выражение для вычисления суммы	Функция	Исходные данные
9	$z=f(a,b)+ f(a^2,b^2)+ +f(a^2-1,b)+ f(a-b,b)+ +f(a^2+b^2,b^2-1)$	$f(u,t)=\begin{cases} u+sin(t), \text{если } u>0; \\ u+t, \text{ если } u\le 0. \end{cases}$	a, b
10	$z =f(\sin\alpha,a)+ f(\cos\alpha,a)+ +f(\sin\alpha,a-1)+ +f(\sin\alpha- \cos\alpha,a^2-1)+ +f(\sin^2 \alpha-1, \cos\alpha+a)$	$f(u,t)=\begin{cases} u^2+t^2, \text{если } u>0 \text{ и } t>0; \\ u+t^2, \text{если } u\le 0 \text{ и } t\le 0; \\ u-t, \text{ если } u>0 \text{ и } t\le 0; \\ u+t, \text{ если } u\le 0 \text{ и } t>0. \end{cases}$	a, α
11	$z =f(\sqrt{\lvert x\rvert}, y)+ f(a, b)+ +f(\sqrt{\lvert x\rvert}+1,-y)+ +f(\lvert x\rvert-\lvert y\rvert,x)+ +f(x+y, a+b)$	$f(u,t)=\begin{cases} u+t, \text{если } u>1; \\ u-t, \text{если } 0\le u\le 1; \\ t-u, \text{ если } u<0. \end{cases}$	x, y, a, b
12	$z=f(\sin(x)+\cos(x),x+y)+ +f(\sin(x),\cos(y))+ +f(x-y,x)+ f(\sin^2(x)-2,a)+ +f(a+3,b+1)$	$f(u,t)=\begin{cases} u+2t, \text{если } u\ge 0; \\ u+t, \text{если } u\le -1; \\ u^2-2t+1, \text{ если } -1<u<0. \end{cases}$	x, y, a, b
13	$z= f(a, b)+ f(a^2, b^2)+ +f(a^2-1,b)+ f(a-b, b)+ f(a^2+b^2, b^2-1)$	$f(u,t)=\begin{cases} u+2t, \text{если } u\ge 0; \\ u+t, \text{если } u\le -1; \\ u^2-2t+1, \text{ если } -1<u<0. \end{cases}$	a, b
14	$z =f(\sin\alpha,a)+ f(\cos\alpha,a)+ +f(\sin\alpha,a-1)+ +f(\sin\alpha- \cos\alpha,a^2-1)+ + f(\sin^2 \alpha-1, \cos\alpha+a)$	$f(u,t)=\begin{cases} u+t, \text{если } u>1; \\ u-t, \text{если } 0\le u\le 1; \\ t-u, \text{ если } u<0. \end{cases}$	a, α
15	$z =f(\sqrt{\lvert x\rvert}, y)+ f(a, b)+ +f(\sqrt{\lvert x\rvert}+1,-y)+ +f(\lvert x\rvert-\lvert y\rvert,x)+ + f(x+y, a+b)$	$f(u,t)=\begin{cases} u^2+t^2, \text{если } u>0 \text{ и } t>0; \\ u+t^2, \text{если } u\le 0 \text{ и } t\le 0; \\ u-t, \text{ если } u>0 \text{ и } t\le 0; \\ u+t, \text{ если } u\le 0 \text{ и } t>0. \end{cases}$	x, y, a, b
16	$z= f(\sin(x)+\cos(x), x+y)+ +f(\sin(x),\cos(y))+ + f(x-y,x)+ f(\sin^2(x)-2,a)+ +f(a+3, b+1)$	$f(u,t)=\begin{cases} u+sin(t), \text{если } u>0; \\ u+t, \text{ если } u\le 0. \end{cases}$	x, y, a, b

ЛАБОРАТОРНАЯ РАБОТА №16

Программирование с использованием процедур

Цель работы:

1) Изучение структуры подпрограммы типа процедура.

2) Изучение механизма передачи данных в процедурах.

3) Получение навыков программирования с использованием процедур.

Процедура является одним из видов подпрограмм и имеет структуру аналогичную структуре головной программы: заголовок, раздел описаний и раздел операторов. Текст процедуры в головной программе располагается в разделе описаний, перед разделом операторов, например:

```
PROGRAM LAB;
LABEL...;
VAR ...;
   {текст процедуры}
BEGIN ... END.
```

Процедура может быть записана в *отдельный файл*. Чтобы включить ее в текст головной программы, используют директивы компилятора

{ $I *имя и маршрут доступа к файлу***}**.

Пример.

Пусть подпрограмма решения квадратных уравнений оформлена в виде отдельного файла KVUR.PAS:

```
PROGRAM LAB;
VAR...;
{$I KVUR.PAS}-т.е. включить в текст программы LAB
                  текст  файла KVUR.PAS.
Begin  ....   end.
```

Если подпрограмма оформляется в виде отдельного файла, в конце ее после end ставят точку с запятой.

Структура процедуры имеет следующий вид:

PROCEDURE *имя***(***формальные параметры***)**;
  раздел описаний;
BEGIN
  раздел операторов;
END;

Здесь *имя* - название процедуры, *раздел описаний* аналогичен разделу описаний головной программы, *формальные параметры* - список ключевых переменных с указанием их типа, *фактические параметры* - те данные, с которыми работает программист в конкретной программе. Формальные и фак-

тические параметры могут совпадать. Процедура может быть и без списка формальных параметров.

Пример *заголовка* процедуры:

PROCEDURE PR(X,Y:REAL; Z:Integer);

Параметры могут быть трех типов: входные (заданные значения); выходные (вычисляемые переменные); параметры процедурного типа.

При описании *выходных* параметров перед ними ставится ключевое слово **VAR**. Например, если входными данными будут коэффициенты a, b, c , выходными - X1 и X2:

PROCEDURE KVUR(a,b,c:REAL; VAR X1:X2:REAL);

Процедурные параметры - параметры, которые входят в список фактических или формальных параметров и обозначают имя процедуры или функции. Переменные процедурного типа должны быть специально описаны:

TYPE *имя типа 1*=**PROCEDURE(***формальные параметры***);**
имя типа 2=**FUNCTION(***формальные параметры***):***тип результата***;**
VAR *переменная 1*: *имя типа 1*; *переменная 2*: *имя типа 2*;

Рекомендуется все процедуры и функции, имена которых присваиваются процедурным переменным, транслировать в специальном режиме: перед заголовком такой процедуры или функции ставится директива компилятора **{$F+}**, а после нее - **{$F-}** ,т.е. компилятору дается указание, что имя процедуры в этой части программы должно восприниматься как возможный процедурный параметр.

Вызов процедуры осуществляется указанием ее имени и перечислением фактических параметров:

имя(*фактические параметры***);**

Список фактических параметров должен соответствовать списку формальных параметров по их количеству, типу и последовательности перечисления. Если в заголовке процедуры отсутствуют формальные параметры, например, PROCEDURE BEZPAR; ,то обращение к ней осуществляется командой BEZPAR; . Если в качестве параметров процедуры используются *данные сложных типов* (например, массивы), необходимо в головной программе описать имя типа этих данных (с помощью TYPE), а затем имя типа указывается в списке формальных параметров.

Например, если в качестве входного параметра используется массив A из 5 вещественных элементов, его описывают в головной программе:

TYPE MAS1=array[1...5] of real; ,

а в процедуре указывают, что переменная A относится к типу MAS1

PROCEDURE B(A:MAS1;VAR S:real);

ЗАДАНИЕ

Разработать процедуру преобразования данных в соответствии с вариантом задания. Составить программу решения задачи с использованием разработанной процедуры. Ввод исходных данных осуществить с клавиатуры в диалоговом режиме. Результаты выполнения программы вывести во внешний файл, снабдив их соответствующими комментариями и предусмотрев форматный вывод числовых значений.

1. Написать подпрограмму умножения вектора на вектор скалярно, согласно формуле $(a,b)=\sum_{i=1}^{n} a_i b_i$. Используя её, найти скалярное произведение вектора C на вектор f, если у каждого из них по 7 компонентов.
2. Написать подпрограмму для отыскания суммы и количества отрицательных элементов массива из n элементов. Используя её, найти величины для массива A из 9 членов.
3. Написать процедуру для суммирования элементов двумерного массива m×n. Используя ее, найти сумму элементов массива B размером 2×3.

4. Написать подпрограмму для нахождения координат центра тяжести системы из n материальных точек:

$$x_c=\sum_{i=1}^{n} x_i m_i \Big/ \sum_{i=1}^{n} m_i \quad ; \quad y_c=\sum_{i=1}^{n} y_i m_i \Big/ \sum_{i=1}^{n} m_i \, .$$

Используя её, найти центр тяжести линейного стержня с k сосредоточенными массивами ($k \le 100$).

5. Найти среднее геометрическое массива из n элементов $SRGE=\sqrt{\sum_{i=1}^{n} a_i^2}$. Вычислить, используя эту подпрограмму, среднее геометрическое массива D из 9 элементов.

6. Найти среднее арифметическое массива из n элементов $SRAR=\frac{1}{n}\sum_{i=1}^{n} a_i$. Используя её, вычислить среднее арифметическое массива F из 7 элементов.

7. Написать подпрограмму суммирования двух n-мерных массивов. Используя её, просуммировать массивы A и B из 8 элементов каждый.

8. Написать подпрограмму для отыскания суммы и количества положительных элементов массива из k элементов. Используя её, найти величины для массива B из 12 членов.

9. Написать подпрограмму для накопления суммы всех отрицательных элементов двумерного массива k×l .Используя эту подпрограмму , просуммировать все элементы массива C размером 5×3.

10. Написать подпрограмму для отыскания количества отрицательных элементов двумерного массива m×n. Используя эту подпрограмму , найти количество отрицательных элементов массива F размером 3×4.

ЛАБОРАТОРНАЯ РАБОТА №17

Передача значений массивов в процедурах

Цель работы:

1) *Изучение механизма передачи параметров в процедурах.*
2) *Получение навыков в написании программ с использованием передачи значений массивов в процедурах.*

Для выполнения лабораторной работы студент должен уметь описывать массивы, знать простейшие алгоритмы нахождения максимального и минимального элементов, знать структуру и порядок оформления процедур.

Если в качестве исходной информации в процедуру передается массив, то его следует передавать по ссылке для экономии памяти, так как в этом случае при вызове процедуры не образуется локальный массив.

Несмотря на то, что обрабатываемые массивы имеют разную длину, они описываются в программе как массивы одного и того же типа, так как при обращении к процедуре типы соответствующих формальных и фактических параметров должны совпадать.

ЗАДАНИЕ

Дано несколько одномерных чисел. Требуется в каждом массиве найти наибольший и наименьший элементы и напечатать их, затем все компоненты каждого массива возвести в квадрат и снова найти наибольший и наименьший элементы. Вычисление максимальной и минимальной величин оформить в виде процедуры, глобальные параметры в процедуре не использовать. Ввести и обработать массивы, длины которых заданы в варианте.

Варианты заданий:

1. Два массива, содержащие 5 и 8 вещественных компонентов.
2. Три массива, содержащие 3,6 и 8 целых компонентов.
3. Четыре массива, содержащие 4,6,3 и 10 целых компонентов.
4. Два массива, содержащие 4 и 6 вещественных компонентов.
5. Три массива, содержащие 5,10 и 4 целых компонента.
6. Четыре массива, содержащие 3,5,8 и 6 вещественных компонентов.
7. Два массива, содержащие 7 и10 вещественных компонентов.
8. Три массива, содержащие 5,4 и 7 целых компонентов.
9. Четыре массива, содержащие 7,3,5 и 4 целых компонента.
10. Два массива, содержащие 6 и 8 вещественных компонентов.

ЛИТЕРАТУРА

1. Офицеров Д.В., Старых В.А. Программирование в интегрированной среде Турбо-Паскаль. Справ. пособие. -Мн, "Беларусь", 1992.-240 с.

2. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. -Томск: МП"РАСКО", 1991.-272 с.: ил.

3. Фигурнов В.Э. IBM PC для пользователя, изд.3-е, исправл. и дополн.- Уфа, Партнерская компания "Дегтярев и сын", НПО "Информатика и компьютеры", 1993.-330 с.: ил.

4. Климов Ю.С., Касаткин А.И., Мороз С.М. Программирование в среде Turbo Pascal 6.0: Справ.пособие. - Мн.: Выш.шк., 1992.-158 с.: ил.

5. Скляров В.А. Программное и лингвистическое обеспечение персональных ЭВМ. Системы общего назначения: Справ. пособие. -Мн, Выш.шк., 1992.-462 с.: ил.

6. Офицеров Д.В., Долгий А.Б., Старых В.А. Программирование на персональных ЭВМ: Практикум: Учеб. пособие. - Мн, Выш.шк., 1993.-256 с.

7. Васюкова Н.Д., Тюляева В.В. Практикум по основам программирования. Язык Паскаль: Учеб. пособие. - М. :, Высш.шк., 1991.-160 с.: ил.

8. Фурунжиев Р.И. Вычислительная техника: практикум.: Учеб. пособие для втузов. - Мн, Выш.шк., 1985.-254 с.: ил.

С о д е р ж а н и е

Учебное издание

ЖИЛЕВИЧ Михаил Иванович
ФИЛИПОВА Людмила Геннадьевна

ИНФОРМАТИКА

Учебно-методическое пособие
к выполнению лабораторных работ
для студентов специальности 1-36 01 07
"Гидропневмосистемы мобильных и
технологических машин"