ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение
высшего профессионального образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Основы программирования на языке Паскаль

Методические указания к лабораторным работам по курсу "Программирование и основы алгоритмизации'

Пенза
Издательство ПГУ
2010

УДК 681.3

П

Основы программирования на языке Паскаль: методические указания / сост.: М. П. Берестень. – Пенза : Изд-во ПГУ, 2010. – 90 с.

Приведены основные принципы построения алгоритмов; дано описание средств языка Паскаль, используемых в лабораторных работах; рассмотрены примеры типовых задач; приведены варианты заданий.

Методические указания подготовлены на кафедре «Автоматика и телемеханика» и предназначены для студентов специальности 210201.

Р е ц е н з е н н т начальник НИС ПНИЭИ С.В. Огурцов

© ГОУ ВПО "Пензенский государственный университет", 2010

Лабораторная работа № 1

Работа в среде Turbo Pascal

Цель работы: изучение основных приемов работы в среде Turbo Pascal.

Основные функции среды Turbo Pascal

Система программирования **Turbo Pascal** предназначена для разработки программ на языке Паскаль и представляет собой совокупность компилятора языка программирования Паскаль и инструментальной программной оболочки.

После запуска Turbo Pascal на экране монитора появляется инструментальная оболочка системы, предоставляющая программисту средства для разработки, отладки и выполнения программы на языке Паскаль. В верхней строке оболочки находится меню режимов работы Turbo Pascal, в нижней строке - краткая справка о назначении основных функциональных клавиш. Центральная часть экрана (заключена в рамку) является окном текстового редактора, где программист осуществляет ввод и редактирования текста программы. Редактор является многооконными и позволяет открывать одновременно несколько окон, в каждом из которых можно редактировать свой файл. Номер окна редактирования (для первых девяти файлов) находится в верхнем правом углу. В верхней строке окна указывается имя файла на диске, в котором хранится редактируемый текст (новым файлам по умолчанию присваиваются имена NONAME00.PAS, NONAME01.PAS и т.д.). Программист, используя меню, может также отобразить на экране вспомогательные окна: отладочного режима (Watch), вывода результатов работы программы (Output), справочной службы (Help) и другие.

Переход в главное меню можно осуществить с помощью мыши или нажатием клавиши F10 на клавиатуре, после чего поместить курсор на нужный пункт и нажать ENTER. В появившемся ниспадающем меню выбрать необходимую функцию и нажать ENTER.

Рассмотрим наиболее востребованные функцию меню.

Пункт меню **File** содержит функции:

New - создание нового документа (при этом открывается новое окно);

Open ... – чтение текста программы из файла;

Save - сохранение текста программы в файл (сохраняется текст из текущего окна);

Save As ... - сохранение текста программы в файл под новым именем;

Change dir ... - смена текущей папки (которую Turbo Pascal использует для сохранения файлов).

Пункт меню **Edit** содержит функции для редактирования текста:

Cut – вырезать;

Copy – копировать;

Paste – вставить.

Пункт меню **Compile** содержит функцию **Compile**, с помощью которой выполняется компиляция программы (при этом производится поиск ошибок в программе).

Пункт меню **Run** содержит функцию **Run**, с помощью которой программа запускается на выполнение.

Для быстрого управления средой Turbo Pascal программист может использовать функциональные клавиши (**F1**,..., **F12**) или комбинации клавиш, за которыми закреплены следующие действия:

F1 - вызвать справку;

F2 - сохранить программу на диск;

F3 - открыть программу с диска в окне редактора;

F5 - развернуть активное окно на весь экран;

F9 - компилировать программу, но не выполнять ее;

F10 - перейти в главное меню;

Ctrl+F9 - компилировать программу, загрузить ее в оперативную память и выполнить;

Alt+F5 – перейти в окно вывода результатов программы;

Alt+X - выход из Turbo Pascal.

Текстовый редактор Turbo Pascal предназначен для создания и редактирования текста программы. Система команд, имеющаяся в редакторе, реализована с помощью закрепленных клавиш или их сочетаний. Некоторые команды приведены ниже:

BACKSPACE - стереть символ слева от курсора;

DEL - стереть символ, на который указывает курсор;

CtrL+Y -стереть строку, на которой располагается курсор;

ENTER - вставить новую строку;

Ctrl+K B - отметить начало блока;

Ctrl+K K - отметить конец блока;

Ctrl+K Y - стереть блок;

Ctrl+K C - копировать блок;

Ctrl+K V - переместить блок;

Ctrl+K H - отменить блок;

Shift+ ← ↑ → ↓ **HOME END** - быстро создать блок;

Ins – переключение режима Вставка/Замена.

При работе в среде Turbo Pascal при разработке небольших программ обычно реализуется следующий порядок действий. В текстовом редакторе формируется текст программы. Если текст готов и записан в файл на диске, то выполняется его чтение (**File | Open** ...). Затем программу необходимо откомпилировать (**Alt+F9** или **Ctrl+F9**). Если в программе есть синтаксические ошибки, то сообщение о первой из них появится в первой строке редактора (при этом она будет выделена красным цветом), а курсор будет помещен на ту строку программы, где обнаружена ошибка. Ошибку следует исправить и программу повторно откомпилировать. Если ошибок не обнаружено, то появится окно, содержащее соответствующее сообщение. Теперь программу можно выполнить (**Run | Run** или **Ctrl+F9**). Отметим, что команда **Ctrl+F9** позволяет откомпилировать программу и, если ошибки отсутствуют, направить ее на выполнение.

В ходе выполнения программы на экране отображается окно вывода результатов программы. Здесь же программист осуществляет ввод с клавиатуры необходимых данных (если это предусмотрено в программе). После завершения работы программы осуществляется возврат в окно редактора. Просмотр результатов в окне вывода результатов программы можно осуществить с помощью команды **Alt+F5**.

Если в ходе выполнения программы обнаружена ошибка, то среда Turbo Pascal прекращает дальнейшие действия программы, восстанавливает окно редактора и помещает курсор на ту строку программы, на которой обнаружена ошибка. Причина прекращения выполнения программы указывается в верхней строке редактора. Ошибку следует исправить и программу повторно выполнить.

Среда Turbo Pascal предоставляет в распоряжение программиста разнообразные средства отладки программ. Приведем наиболее востребованные:

F7 – пошаговое исполнение программы;

Ctrl+F4 - просмотр значения переменной, на которую вданный момент указывает курсор.

Среда Turbo Pascal имеет справочную систему, воспользоваться которой можно, нажав клавишу **F1**. Информация об элементе программы, на который указывает в данный момент курсор, может быть получена при нажатии комбинации клавиш **Ctrl+F1**.

Структура программы на языке Паскаль

Программа, написанная на языке Паскаль, состоит из заголовка, описательной части и операторной части. Кратко рассмотрим структуру программы на простейшем примере, который реализует вычисление произведения двух чисел.

```
Program ABC; {заголовок программы}
Var
    a, b, s: integer; {описательная часть программы}
BEGIN {начало операторной части программы}
    Write('a='); {вывод на экран сообщения a=}
    Readln(a); {ввод числа a}
    Write('b='); {вывод на экран сообщения b=}
    Readln(b); {ввод числа b}
    s:=a*b; {вычисление произведения s}
    Writeln('s=',s); {вывод результата}
END. {конец операторной части программы}
```

Заголовок программы имеет вид:

Program <*идентификатор программы*>;

(символы <...> являются элементами описания формата и в программу не переносятся). Идентификатор – это имя программы. В приведенном примере идентификатор программы – **ABC**. Идентификаторы также используются для обозначения переменных, меток, констант и других элементов программы. Идентификатор может состоять из латинских букв, цифр и символа подчеркивания. Начинается идентификатор с буквы или символа подчеркивания. Длина идентификатора не может быть больше 127 символов, причем значащими являются только первые 63. Прописные и строчные буквы в идентификаторах не различаются.

В описательной части блока определяются все идентификаторы, которые будут использованы в операторной части. Описательная часть включает несколько разделов в соответствии с видами элементов, имеющимися в Паскале: метки, константы, типы данных, переменные, функции и процедуры. Все эти разделы не являются обязательными и, т.о., описательная часть блока может полностью отсутствовать. В примере используются только переменные, поэтому в описательной части только один раздел – раздел описания переменных. Он начинается со слова **var,** за которым следует список перечисляемых через запятую переменных, затем символ двоеточие и указывается тип переменных. В примере все переменные (**a, b, s**) целого типа (**integer**), то есть их значениями могут быть только целые числа.

Операторная часть начинается с зарезервированного слова **BEGIN** и заканчивается зарезервированным словом **END,**. вслед за которым ставится точка. Операторная часть содержит операторы, которые выполняются в естественном порядке, то есть по очереди как они записаны, начиная с оператора, стоящего после слова **BEGIN**. В примере первый оператор - **write –** оператор вывода (точнее стандартная процедура вывода). При запуске программы он выведет на экран сообщение **a=**. Следующий оператор - **readln** – оператор ввода. Когда наступает очередь этого оператора, выполнения программы приостанавливается – она ждет, когда с клавиатуры будет введено значение переменной, указанной в списке ввода (в примере – переменная **a**). На клавиатуре следует набрать значение переменной **a** и нажать клавишу Enter. Следующая пара операторов обеспечит ввод значения переменной **b**. Далее стоит оператор присваивания (**:=** - знак присваивания), после выполнения которого переменная **s** получит значение равное произведению двух чисел **a** и **b**. Результат будет выведен на экран с помощью оператора **writeln**. После выполнения оператора **writeln**, в отличие от оператора **write**, курсор автоматически переводится на следующую строку.

Порядок выполнения работы

1. Изучить основные функции среды Turbo Pascal.
2. Изучить структуру программы на Паскале.
3. Разработать программу на языке Паскаль для вычисления периметра треугольника, если известны длины его сторон.
4. Подготовить отчет.

Требования к оформлению отчета (для всех лабораторных работ курса).

Отчет оформляется на листах формата А4 в машинописном или рукописном виде и должен содержать следующие элементы:

- титульный лист;
- задание (с обязательным указанием варианта);
- описание метода решения (необязательный элемент);
- описание алгоритма решения задачи (в виде блок-схемы);
- листинг программы на языке Паскаль;
- тестовые примеры;
- список использованной литературы (необязательный элемент).

Лабораторная работа № 2

Программирование процессов линейной структуры

Цель работы: изучение основных приемов проектирования и отладки программ линейной структуры на Паскале.

Алгоритмы и способы их представления

Алгоритмом называется точное предписание, задающее преобразование исходных данных в искомый результат. При этом могут производиться числовые вычисления, преобразования буквенных выражений, некоторых символов и т.д.

Числа, буквы, символы, над которыми производятся те или иные операции (вычисления, преобразования и т.д.), называются ***операндами***, а указания, предписания, правила преобразования операндов - ***операторами***.

В процессе разработки алгоритма могут использоваться различные способы его описания, отличающиеся по простоте, наглядности, компактности, степени формализации, ориентации на машинную реализацию и другим показателям. В практике программирования наибольшее распространение получили словесная запись алгоритмов, схемы алгоритмов, структурограммы (диаграммы Насси - Шнейдермана).

Схема алгоритма - это графическое представление алгоритма, дополненное элементами словесной записи. Каждый пункт алгоритма отображается на схеме некоторой геометрической фигурой - блоком (блочным символом), причем различным по типу выполняемых действий блокам соответствуют различные геометрические фигуры, изображаемые в соответствии с ГОСТом. Правила выполнения схем алгоритмов регламентирует ГОСТ 19.701-90.

Схема программы состоит из:

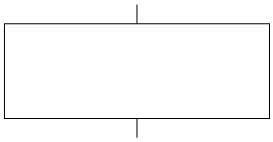
1) символов процесса, указывающих фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);

2) линейных символов, указывающих поток управления;

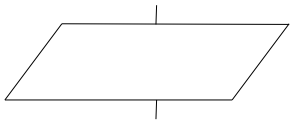
3) специальных символов, используемых для облегчения написания и чтения схемы.

Ниже приведены некоторые наиболее часто используемые символы с указанием назначения.

Символы процесса:



процесс - отображает функцию обработки данных любого вида;



данные - отображение данных, носитель не определен;

Линейные символы:

линия – символ отображает поток данных или управления.

При необходимости или для повышения удобочитаемости могут быть добавлены стрелки-указатели.

Специальные символы:

соединитель – отображает выход из одной части схемы и вход в другую часть этой схемы и используется для обрыва линии и продолжения ее в другом месте.

Соответствующие символы - соединители должны содержать одно и тоже уникальное обозначение.

терминатор - отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы алгоритма, внешнее использование и источник или пункт назначения данных).

Графические символы на схемах соединяются линиями потока информации. Основное направление потока информации идет сверху вниз и слева направо (стрелки на линиях могут не указываться). В других случаях применение стрелок обязательно. По отношению к блоку линии потока могут быть входящими или выходящими.

Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линия должна быть направлена к центру символа.

Алгоритм любой степени сложности можно построить с помощью основного базового набора структур:

1) последовательная (линейная) структура;
2) ветвящаяся структура;
3) циклическая структура.

Процессы линейной структуры

Наиболее простыми для понимания и использования являются линейные структуры. Линейным называется алгоритм (фрагмент алгоритма), в котором отдельные предписания выполняются в естественном порядке (в

порядке записи) независимо от значений исходных данных и промежуточных результатов.

Рассмотрим задачу вычисления площади треугольника по трем сторонам (a, b, c). Словесная запись алгоритма решения этой задачи может быть представлена следующим образом:

Пункт 1. Вычислить полупериметр треугольника $p = (a + b = c)/2$ и перейти к пункту 2.

Пункт 2. Вычислить площадь треугольника $s = \sqrt{p(p-a)(p-b)(p-c)}$ и перейти к пункту 3.

Пункт 3. Вывести на экран результат s и завершить вычисления.

Схема алгоритма решения этой задачи представлена на рисунке 2.1.

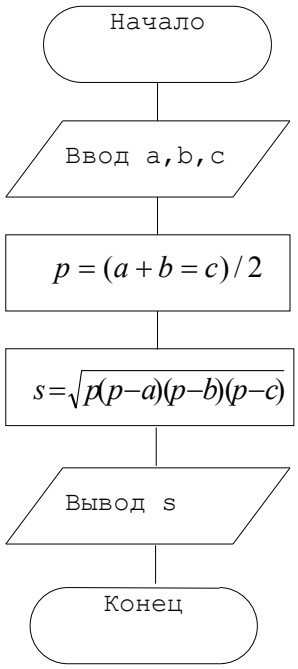


Рисунок 2.1

Полученный алгоритм является линейным, т.к. все предписания выполняются в порядке записи, начиная с блока "Начало" и заканчивая блоком "Конец". Отметим, что все предписания (блоки) будут выполнены, причем только один раз.

Ниже приведена программа на языке Паскаль, реализующая этот алгоритм.

```
Program P201;
var
    a, b, c, p, s: real;
```

```
begin
        readln(a, b, c);
        p:= (a+b+c)/2;
        s:= sqrt(p*(p-a)*(p-b)*(p-c));
        writeln('s=', s);
end.
```

Основы языка Паскаль

Рассмотрим на примере некоторые элементы языка Паскаль, необходимые для выполнения данной лабораторной работы.

Константы. Под константой понимается данное, значение которого в процессе обработки постоянно и не может быть изменено. Обычно константа определяется непосредственно там, где она используется, путем явного задания ее значения. В примере только одна константа – 2 (в операторе p:= (a+b+c)/2;). В Паскале, кроме того, разрешается присваивать константам имена и использовать эти имена в нужном месте программы вместо соответствующей константы. Такие "именованные" константы определяются в программе на Паскале в разделе определения констант (раздел const).

В Паскале стандартно определены две категории констант: *числовые* и *строковые*. Их значения соответствуют стандартно существующим в Паскале типам данных.

Рассмотрим основные виды числовых констант.

Целочисленные константы задают значения целого типа и могут быть представлены в десятичной или шестнадцатеричной форме. Признаком шестнадцатеричной формы записи служит знак $, который предшествует числу. Значение целочисленной константы должно находиться в диапазоне от минус 32768 до плюс 32767 в случае десятичной формы представления и в диапазоне от $0000 до $FFFF в случае шестнадцатеричной формы представления.

Примеры целочисленных констант:

```
1
12345
-1
$51
```

Вещественные константы задают значения вещественного типа и могут быть представлены в форме с фиксированной или с плавающей точкой. В форме с фиксированной точкой вещественная константа представ-

ляет собой последовательность десятичных цифр с десятичной точкой, разделяющей целую и дробную части числа. Целой части может предшествовать знак.

Примеры вещественных констант:
1.0
123.4567
-0.0327

В форме с плавающей точкой число состоит из мантиссы, записываемой в виде целого числа или вещественного числа с фиксированной точкой и порядка, записываемого в виде целого числа со знаком. Между мантиссой и порядком записывается буква E, которая читается как "умножить на 10 в степени". В мантиссе должно быть не более 11 значащих цифр. Значение вещественного числа должно находиться в диапазоне от 1E-38 до 1E+38.

Примеры приведены в таблице 2.1.

Таблица 2.1

форма с плавающей точкой	форма с фиксированной точкой
12E3	12000
7E-02	0.07
1.1234567809E+5	-112345.67809

Строковая константа задает значение строкового типа и представляет собой заключенную в апострофы последовательность любых символов из набора символов машины. Значением константы является указанная последовательность (строка) символов, исключая начальный и конечный апострофы. Для включения апострофа в значение константы используются два следующих друг за другом апострофа.

Пример:
'PASCAL'
'ПАСКАЛЬ'
'A"B'

Переменная. Переменная является основным элементом данных в Паскале. Переменные в ходе выполнения программы могут менять свое значение в пределах данного типа.

Каждая переменная должна быть определена перед ее использованием в программе. При определении переменной задается ее идентификатор и тип значений, который она может принимать. Переменные определяются в разделе определений переменных, который начинается с зарезервирован-

ного слова **var** и содержит список определений. Каждый элемент списка определяет одну или несколько переменных некоторого типа. Идентификаторы переменных разделяются запятыми, а тип отделяется от них двоеточием. Тип можно задавать либо множеством значений, либо идентификатором, предварительно определенным в разделе определений типов. Каждый элемент списка заканчивается точкой с запятой.

Пример:

```
var
    a, b: real;
    I, J, K: integer;
    Period: DAY;
    X: array [1..128] of Byte;
```

Описание переменной не задает каких-либо значений переменной.

Тип. В языке Паскаль могут использоваться данные, обладающие разными свойствами, т.е. разных типов. Концепция типов данных в паскале характеризуется следующими основными свойствами:

- тип является атрибутом, присущим любой константе, переменной, выражению или функции;
- тип определяет множество значений, которые могут принимать объекты этого типа;
- тип определяет множество операций, допустимых над значениями этого типа;
- тип определяет внутреннюю форму представления данных в компьютере.

Различаются **простые** (скалярные) и **сложные** (структурированные) типы данных. *Простыми типами* считаются такие типы, значения которых не распадаются на компоненты. К простым типам относятся стандартные типы и типы, определяемые пользователем.

Для сложных типов характерна структурность значений. Значения *сложных типов* представляют собой совокупность компонент, каждая из которых является значением некоторого другого типа.

В Паскале имеется набор стандартно определенных простых типов данных. С этими типами связаны стандартные идентификаторы, и они не требуют описания в разделе определений типов.

Тип "целое" обозначается стандартным идентификатором Integer и представляет множество целых чисел в пределах от минус 32768 до плюс 32767.

Тип "вещественное" обозначается стандартным идентификатором Real и представляет множество вещественных чисел в пределах от +1E-38 до 1E+38 с мантиссой, содержащей от 1 до 11 значащих цифр.

Тип "логическое" обозначается стандартным идентификатором Boolean и включает два значения, обозначаемые стандартными идентификаторами False (ложь) и True (истина), причем значение False предшествует значению True.

Тип "символьное" обозначается стандартным идентификатором Char и представляет множество символов из набора символов компьютера, упорядоченных в соответствии с их кодами в таблице кодов компьютеров. Порядковые номера значений находятся в диапазоне от 0 до 255.

Выражения. Основными объектами, из которых конструируется исполняемая часть программы на Паскале, являются константы, переменные и обращения к функциям. Каждый из этих объектов характеризуется своим значением и принадлежит к какому-либо типу данных. С помощью знаков операций и скобок из них можно составлять выражения, которые фактически представляют собой правила получения новых значений.

Выражение - это формула для вычисления значения. В общем случае выражение представляет собой комбинацию переменных, констант и обращений к функциям, соединенных знаками операций. Порядок выполнения операции при вычислении выражения определяется их приоритетами и может регулироваться с помощью скобок. В простейшем случае выражение может состоять только из переменной или константы или обращения к функции.

Каждая операция (функция) требует операндов фиксированного типа и выдает результат фиксированного типа. Как правило, операция выполняется над операндами одинакового типа. Если операция допускает совместное использование операндов различных типов (операнд целого и вещественного), то специально оговаривается.

Операции делят на пять групп. Перечислим их в порядке приоритета их выполнения:

- операция изменения знака;
- операция логического отрицания;
- операции типа умножения;
- операции типа сложения;
- операции отношения.

Операция изменения знака выдает в качестве результата значение указанного операнда с обратным знаком. Операнд может быть целого или вещественного типа.

Пример:

-2

-A .

В Паскале в выражении не могут следовать подряд два знака операции. **Исключение:** 2+-5.

Операция логического отрицания обозначается зарезервированным словом NOT и имеет один операнд логического типа. Результатом операции является значение логического типа, противоположное значению операнда

Пример:

NOT True
NOT x

Операции типа умножения допускают операнды целого, вещественного и логического типов (таблица 2.2).

Таблица 2.2

Знак операции	Операция	Типы операндов	Тип результата
*	умножение	целое целое, веществ. вещественное	целое вещественное вещественное
/	деление	целое целое, веществ. вещественное	вещественное вещественное вещественное
DIV	целочисленное деление	целое	целое
MOD	деление по модулю	целое	целое
AND	логическое умножение (конъюнкция)	логическое	логическое

Примеры.

Выражение	Результат	
2*3	6	
5/2	2.5	
5 DIV 2	2	
9 MOD 4	1	
A*-B	-6	(при A=2 и B=3)
True AND False	False	

Операции типа сложения допускают операнды целого, вещественного и логического типов (таблица 2.3).

Таблица 2.3

Знак операции	Операция	Типы операндов	Тип результата
+	сложение	целое	целое
-	вычитание	целое, веществ.	вещественное
OR	логическое сложение (дизъюнкция)	логическое	логическое
XOR	сложение по модулю два	логическое	логическое

Примеры:

Выражение	Результат
2+2	4
2.+2	4.0
True OR False	True
True XOR False	True

Операции отношения допускают операнды любого простого типа. Результат операции отношения всегда имеет логический тип (таблица 2.4).

Таблица 2.4

Знак операции	Операция
=	равно
<>	не равно
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

Примеры:

Выражение	Результат
2>3	False

Стандартные функции языка Паскаль. В выражениях часто используются стандартные функции для вычисления математических функций (таблица 2.5). Как и все функции Паскаля они имеют следующий формат:

<идентификатор функции>(<список параметров>).

Таблица 2.5

Функция	Обозначение	Тип аргумента	Тип функции	Ограничения на значения аргумента
x^2	sqr(x)	целое вещественное	целое вещественное	
$\sqrt{x}$	sqrt(x)	целое вещественное	вещественное	$x \geq 0$
$\sin x$	sin(x)	целое вещественное	вещественное	аргумент в радианах
$\cos x$	cos(x)	целое вещественное	вещественное	аргумент в радианах
e^x	exp(x)	целое вещественное	вещественное	
$\ln x$	ln(x)	целое вещественное	вещественное	$x>0$
$\|x\|$	abs(x)	целое вещественное	целое вещественное	
arctg x	ArcTan(x)	целое вещественное	вещественное	результат в радианах в диапазоне $-\pi/2$ до $\pi/2$

Функция **Pi** (без параметров) возвращает значение числа **ПИ**.

Операция возведения в степень в Паскале отсутствует. Поэтому, если показатель степени число целое, то возведение в степень выполняется простым перемножением:

a^3: a*a*a.

При больших значениях целой степени используется оператор цикла.

Если показатель степени рациональный, то a^x запишется exp(ln(a)*x) (a должна быть больше 0).

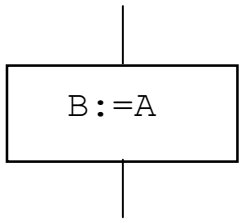
В выражениях сначала вычисляются функции, а затем выполняются операции в соответствии со своими приоритетами.

Для регулирования порядка действий в Паскале используются *скобки*, причем только круглые. Число открывающихся и закрывающихся скобок должно совпадать. Смысл скобок тот же, что и в математике.

Пример (порядок выполнения операций указан над знаком операции).

```
  2  1
a*(x+y)
  1      3     2
(a>b) AND (x=0)
```

Оператор присваивания. Оператор присваивания служит для присваивания переменной некоторого значения. Он состоит из идентификатора переменной, за которым следует специальный символ ":=", а затем выражение, значение которого должно быть присвоено переменной. Значение можно присваивать переменной любого типа. Необходимо, чтобы тип значения выражения совпадал с типом переменной. Как исключение, переменной вещественного типа можно присвоить целочисленное значение.



В схеме алгоритма оператору присваивания соответствует символ процесс.

Примеры:

```
L:=False;
C:=2*Pi*R;
N:=(A='0') AND (B<>'1')
```

Порядок выполнения работы

1. Изучить приемы проектирования программ линейной структуры.
2. Изучить следующие элементы языка Паскаль: константы, переменные, стандартные типы, выражения, оператор присваивания.
3. Разработать алгоритм и программу на языке Паскаль для вычисления арифметического выражения (в соответствии с вариантом).
4. Подготовить отчет.

Варианты заданий

Примечание.

Результат может быть получен не для всех возможных значений исходных данных.

2.1. Вычислить y для произвольных значений a, b, x

$$y = \frac{(\sin^2 x + \cos^2(x-a))(x+a^3)}{e^b - 1}.$$

2.2. Вычислить y для произвольных значений a, b, x:

$$y = \frac{(a + b\sin x)\cos x}{tgx + 1} - x^3$$

2.3. Вычислить y для произвольных значений a, b, x:

$$y = \frac{e^x(a + b\,tg\,x)}{e^{a+2b}} + 2$$

2.4. Вычислить y для произвольных значений a, b, x:

$$y = \frac{e^{a+x^2}\left|5 + abx^2\right|}{2 + \sin(x+1)}$$

2.5. Вычислить y для произвольных значений a, b, x:

$$y = \sqrt{\frac{x^2 + (a+b)^2}{abx^3}}\,.$$

2.6. Вычислить y для произвольных значений a, b, x:

$$y = \frac{x^{\frac{3}{2}} + \sin(a+b)^2}{(2+x)^3} - 1$$

2.7. Вычислить y для произвольных значений a, b, x:

$$y = \frac{5x - \sqrt{2 + \sin^2(x+a)}}{b^2 - tgx} + \frac{a}{1+4b}$$

2.8. Вычислить y для произвольных значений a, b, x:

$$y = ((1+x)\sin x + 2ab^2) + \frac{a+b}{x+2ab}.$$

2.9. Вычислить y для произвольных значений a, b, x:

$$y = tg(x + ab^2) + \sqrt{x^2+1} - \frac{e^{x^2+1}}{a+b}$$

2.10. Вычислить y для произвольных значений a, b, x:

$$y = \frac{\left|x^2 - ab^3\right|}{\sqrt{x}} + \frac{ab}{x^2+a}.$$

2.11. Вычислить y для произвольных значений a, b, x:

$$y = \frac{x^4 + a^{\frac{3}{2}}}{\sqrt{x^2+1}} - \sqrt{x^3 + 2ab.}$$

2.12. Вычислить y для произвольных значений a, b, x:

$$y = tg(x^2 + \frac{1}{a}) - \frac{\frac{1}{x^2} - 2ab}{4a + 7}.$$

2.13. Вычислить y для произвольных значений a, b, x:

$$y = \frac{e^{x+a^2} - \frac{1}{b}}{\sqrt{x+a}} - \frac{a+b}{x^3 + tg\,x}.$$

2.14. Вычислить y для произвольных значений a, b, x:

$$y = \frac{e^{xb^2} + tg(x+a^2)}{(a+b)^3 - 5} + \frac{1}{a+b}.$$

2.15. Вычислить y для произвольных значений a, b, x:

$$y = \frac{tg(x+1) + \sin x^2}{\cos^2 x - 1} + \frac{a+b}{a^2 + b^2}.$$

2.16. Вычислить y для произвольных значений a, b, x:

$$y = \frac{\sin^2(x + 2ab) - \cos(\sqrt{x} - a^2 b^2)}{x^2 + \sin^2(b + a^2)}.$$

2.17. Вычислить y для произвольных значений a, b, x:

$$y = \frac{5x - \sqrt{2+x}}{b^2 - ctg\,x} + \frac{\ln a}{1 + 4b}$$

2.18. Вычислить y для произвольных значений a, b, x:

$$y = ((1+x)\sin\frac{x}{2a+1} + 2b^2) + \frac{2a+b}{x + 2ab}.$$

2.19. Вычислить y для произвольных значений a, b, x:

$$y = ctg(\frac{x}{a+1} + b^2) + \sqrt{x^2 - 1} - \frac{e^{x^2+1}}{a - 3b}$$

2.20. Вычислить y для произвольных значений a, b, x:

$$y = \frac{\left|x^2 - ab^3\right|}{\sqrt{x}} + \frac{\sin b}{\sqrt{x^2 + a}}.$$

2.21. Вычислить y для произвольных значений a, b, x:

$$y = \frac{4x^4 + a^{\frac{3}{2}}}{\sqrt{x^2 + 1}} \sqrt{x^3 + 2(a - b.}$$

2.22. Вычислить y для произвольных значений a, b, x:

$$y = tg(ax^2 + \frac{b}{a+1}) - \frac{\frac{b+a}{x^2} - 2ab}{4a+1}.$$

2.23. Вычислить y для произвольных значений a, b, x:

$$y = \frac{e^{x+a^2} - \frac{1}{b}}{\sqrt{4x + a}} - \frac{3a + b}{x^3 + tg\frac{x}{a+b}}.$$

2.24. Вычислить y для произвольных значений a, b, x:

$$y = \frac{3e^{xb^2} + 3tg(x + a^2)}{(3a + b)^3 - 1}.$$

2.25. Вычислить y для произвольных значений a, b, x:

$$y = \frac{tg(3x + 1) + \sin x^2}{2\cos^2 x - 1} + \frac{a + 2b}{a^2}.$$

2.26. Вычислить y для произвольных значений a, b, x:

$$y = \frac{\cos^2(x + 2ab) - tg(\sqrt{x} - a^2 b^2)}{3x^2 + \sin^2(b + a)}.$$

2.27. Вычислить y для произвольных значений a, b, x:

$$y = \frac{\frac{5x}{a+b} - \sqrt{2 + x}}{b^2 - ctg\,x}$$

2.28. Вычислить y для произвольных значений a, b, x:

$$y = ((1 + a^x)\sin\frac{x - a}{2a + 1} + b^2) + \frac{2a + b}{x}.$$

2.29. Вычислить y для произвольных значений a, b, x:

$$y = (\frac{x}{a+1} + b^2) + \sqrt{x^2 - 2a} - \frac{3e^{x^2}}{a - 3b}$$

2.30. Вычислить y для произвольных значений a, b, x:

$$y = 2\frac{\left|x^2 - ab^3\right|}{\sqrt{x+a}} + \frac{\sin b}{\sqrt{x^2+a}}.$$

Лабораторная работа № 3

Программирование процессов ветвящейся структуры

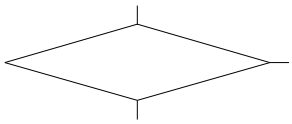
Цель работы: изучение основных приемов проектирования и отладки программ ветвящейся структуры на Паскале.

Процессы ветвящейся структуры

При решении задач часто используются **ветвящиеся структуры**, т.е. такие, в которых в зависимости от исходных данных или промежуточных результатов алгоритм реализуется по одному из нескольких, заранее предусмотренных (возможных) направлений. Такие направления часто называются ветвями.

Каждая ветвь может быть любой степени сложности, а может вообще не содержать предписаний, т.е. быть вырожденной. Выбор той или иной ветви осуществляется в зависимости от результата проверки условия с конкретными данными. В каждом случае алгоритм реализуется только по одной ветви, а выполнение других исключается.

Для реализации ветвлений в схемах алгоритмов, используется символ "решение", который отображает решение или функцию переключательного типа. Имеет один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычислений могут быть записаны по соседству с линиями, отображающими эти пути.



Рассмотрим алгоритм вычисление *ctg(x)* (рисунок 3.1).

Алгоритм предписывает ввести с клавиатуры значение переменной ***x*** и вычислить для нее значения синуса и косинуса (промежуточные результаты будут присвоены переменным ***b*** и ***a***). Чтобы исключить недопустимую операцию (деление на 0), проверим значение знаменателя (переменная ***b***). Если b не равна нулю, то вычисляем значение котангенса (переменная ***y***) и выводим его на экран. Если b равна нулю, то на экран будет выведено соответствующее сообщение, а значение котангенса вычислено не будет.

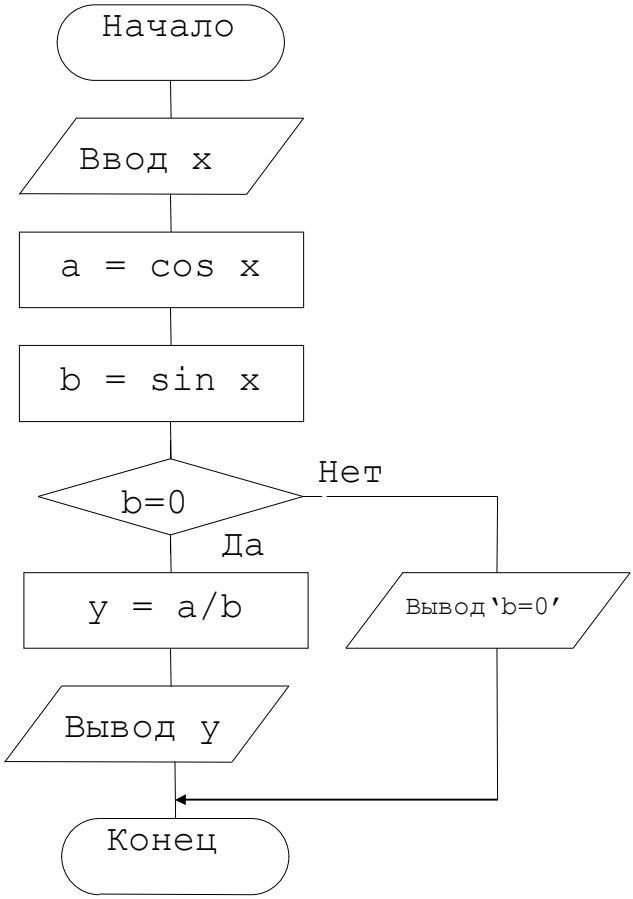


Рисунок 3.1

Ниже приведена программа на языке Паскаль, реализующая этот алгоритм.

```
Program P301;
var
        a, b, x, y: real;
begin
        readln(x);
        a:= cos(x);
        b:= sin(x);
        if b=0 then
                begin
                y:=a/b;
                writeln('y=', y);
                end
                else writeln('b=0');
end.
```

Средства языка Паскаль, используемые при реализации ветвящихся структур

Составной оператор позволяет поместить несколько операторов там, где синтаксис допускает только один оператор. Такая необходимость часто возникает при написании внутренних операторов для таких сложных операторов, как операторы цикла, условные операторы и т.д. Составной оператор начинается с зарезервированного слова begin, заканчивается зарезервированным словом end и включает любое количество операторов, разделенных точкой с запятой. Операторы внутри составного оператора выполняются в той последовательности, в которой они записаны.

Условный оператор (оператор if) определяет выбор и выполнение одного из указанных в нем внутренних операторов.

Формат:

if <логическое выражение> then <оператор>;

или

if <логическое выражение> then <оператор1>
else <оператор2>;

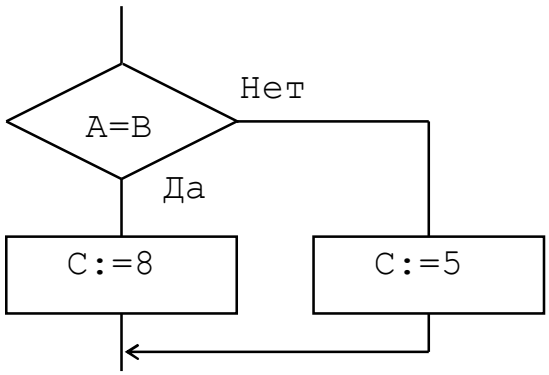
Если указанное логическое выражение истинно, то выполняется оператор, следующий за словом then. В противном случае выполняется оператор, следующий за словом else или, если эта часть оператора отсутствует, следующий за if оператор.

Оператор if в схеме алгоритма соответствует символу "решение" с двумя выходами.

Примеры:

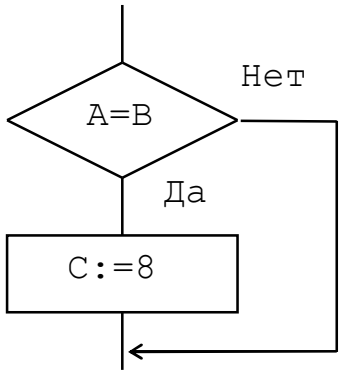
1) Оператор if с альтернативой

```
if a=b
    then c=8
    else c=5;
```



2) Оператор if без альтернативы

```
if a=b then c=8;
```



3) Если после слова then (или else) требуется больше одного оператора, тогда необходимо использовать составной оператор

```
if a>0
        then begin x=y; z=0;end
        else z=2;
```

4) Условные операторы можно вкладывать друг в друга. Если условный оператор содержит другие условные операторы, то слово else соотносится с ближайшим ключевым словом if, еще не связанным ни с каким ключевым словом else.

```
if p
        then
                if r='*'
                        then begin x=y; z=0;end
                        else z=2;

if a
        then
                if b
                        then c:=d
        else c:=1
        else c:=2;
```

При использовании оператора IF важно правильно составить логическое выражение. Чаще всего в качестве логического выражения выступает некоторое условие (a>0 и т.п.), хотя оно может быть представлено и переменной логического типа. Если оператор (или группа операторов) должен быть исполнен только в случае одновременного выполнения нескольких условий (например, a>0 и c=b), то логическое выражение будет иметь вид (a>0) and (c=b). Отметим, что операции отношения имеют самый низкий приоритет, поэтому они должны быть заключены в скобки, чтобы обеспечить необходимый порядок выполнения операций.

Правила вычислений логических выражений приведены в таблице 3.1.

Таблице 3.1

Значение операнда		Значение операции			
x	y	not x	x and y	x or y	x xor y
False	False	True	False	False	False
False	True	True	False	True	True
True	False	False	False	True	True
True	True	False	True	True	False

Оператор перехода указывает, что выполнение программы следует продолжить с заданного меткой оператора. Оператор перехода состоит из зарезервированного слова goto, за которым следует метка. При использовании оператора goto должны выполняться следующие правила:

- метка должна быть описана в разделе описаний меток того блока, в котором она используется (глобальных меток не бывает!);

- поскольку областью действия является блок, в котором она описана, оператор перехода не может передавать управление из одного блока в другой.

Любому оператору в программе на Паскале может предшествовать метка. В качестве меток используются целые числа без знака и идентификаторы. Метка записывается как целое число или идентификатор, за которым следует двоеточие.

Все метки должны быть определены в разделе описания меток. Раздел описания меток начинается с зарезервированного слова label, за которым следует список меток, разделенных запятыми. Раздел заканчивается точкой с запятой.

Пример:

```
Label 1, Stop;
begin
...
1: x:=0;
...
goto Stop;
...
goto 1;
...
Stop: writeln('x=', x);
...
```

В блоке может быть несколько операторов goto, позволяющих перейти к одной и той же метке. В блоке не может быть двух одинаковых меток.

Рассмотрим несколько примеров.

Пример № 1.

Для произвольных значений ***x*** вычислить

$$y = \begin{cases} \ln x, & при\ \mathrm{x} > 0 \\ \sin x, & при\ x \le 0. \end{cases}$$

Даже для этой очень простой задачи может быть предложено несколько вариантов решения. Приведем два из них (рисунки 3.2 и 3.3).

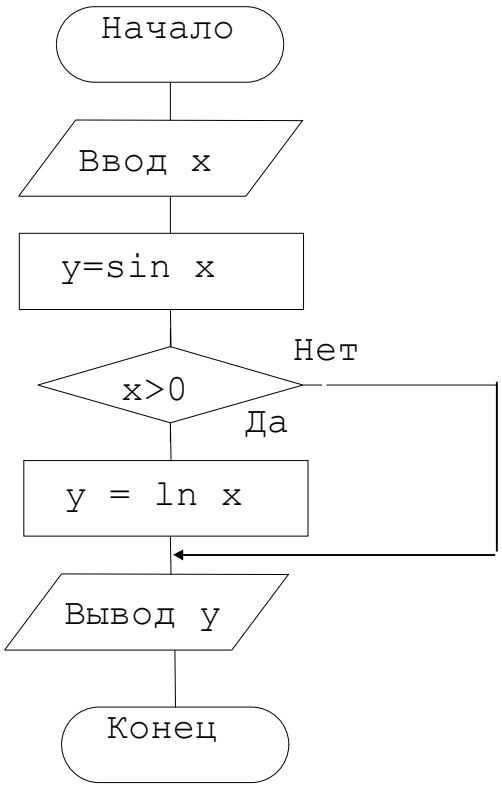
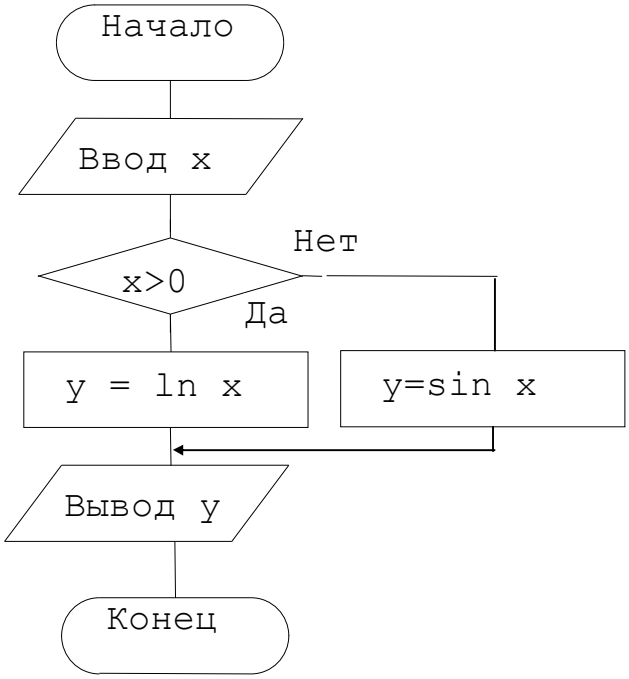


Рисунок 3.2

```
Program P302;
var
        x, y: real;
begin
        readln(x);
        y:= sin(x);
        if x>0 then y:= ln(x);
        writeln('y=', y)
end.
```




```
Program P303;
var
        x, y: real;
begin
        readln(x);
        if x>0
                then  y:= ln(x)
                else  y:= sin(x);
        writeln('y=', y)
end.
```

Рисунок 3.3

Пример № 2.

Принадлежит ли точка с произвольными координатами ***(x,y)*** заштрихованной области (Рисунок 3.4)?

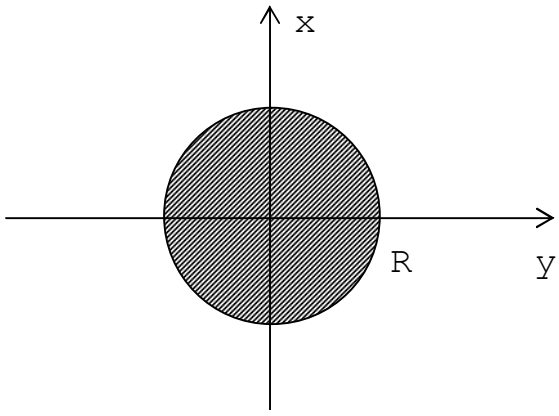


Рисунок 3.4

Плоскость разделена на два непересекающихся множества точек: находящиеся внутри окружности радиуса R с центром в начале координат и все остальные. Поэтому для решения задачи достаточно сформулировать математическое описание одного из этих множеств и выполнить проверку на принадлежность для точки ***(x,y)***. Программа на языке Паскаль может быть такой:

```
Program P304;
var
        x, y, R: real;
begin
        readln(x, y, R);
        if sqrt(sqr(x)+sqr(y))>R
                then  writeln('Не принадлежит')
                else  writeln('Принадлежит');
end.
```

Пример № 3.

Написать программу для вычисления значения ***y***, не используя логические операции.

y:= a or b and c.

Отметим, что все переменные (**y, a, b, c**) должны быть логического типа. Учитывая свойства и приоритет операций логического умножения и сложения, можно отметить, что **y** получит значение TRUE, если значение TRUE имеет переменная a или одновременно переменные **b** и **c**.

Фрагмент программы на языке Паскаль может быть таким:

```
if  a
        then y:=TRUE
        else
                if b
                        then
                                if c then y:=TRUE
                                    else y:=FALSE
                        else y:=FALSE
```

Порядок выполнения работы

1. Изучить приемы проектирования программ ветвящейся структуры.
2. Изучить следующие операторы языка Паскаль: составной, условный и перехода.
3. Разработать алгоритм и программу на языке Паскаль для вычисления арифметического выражения (в соответствии с вариантом).
4. Подготовить отчет.

Варианты заданий

Вариант 3.1

Для произвольного значения x вычислить

$$y=\begin{cases}\dfrac{\sin 2x+x}{2^x} & при\ x>1\\ \dfrac{x+x^2+x^4}{\cos(x+1)} & при\ 1\ge x\ge 0\\ x^2+2 & при\ x<0\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.2

Для произвольных значений x, a, c вычислить

$$y=\begin{cases}\dfrac{5a-2c}{2+\sin 3x} & при\ x>0\ \ и\ \ c>0\\ 2{,}5\sin 3x & при\ x\le 0\\ 2x\sin(x+4) & при\ x>0\ \ и\ \ c\le 0\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.3

Для произвольных значений x, a вычислить

$$y=\begin{cases}\dfrac{\ln(|x|+\cos x)-2x^2}{|1{,}5a|+1} & при\ a>x\\ |2x-a|+2ax & при\ x>a\\ 4ax+2^x & при\ x=a\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.4

Для произвольных значений x, a вычислить

$$y=\begin{cases}\sqrt{25x-a} & при\ x>1\ \ и\ \ a<10\\ x^2+a & при\ x\le 1\\ \dfrac{2\ln x+a\sin x}{a+x} & при\ x>1\ \ и\ \ a\ge 10\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.5

Для произвольных значений *x, a* вычислить

$$y=\begin{cases}\sin x-\cos a & при\ \ a\le -1\\ (\cos a-10^{\sin x})x & при\ \ a\ge 1\\ \sin^2 x\cdot \operatorname{ctg}\dfrac{2+a}{10} & при\ -1<a<1\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.6

Для произвольных значений *x, a* вычислить

$$y=\begin{cases}\dfrac{|x\sin a|}{1+(\ln|x|)^2} & при\ \ ax<0\\ a^x & при\ \ 2\ge ax\ge 0\\ \ln ax+ax-a\sin ax & при\ \ ax>2\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.7

Ввести с клавиатуры значения переменных *A, B, C*. Вывести на экран идентификатор и значение наибольшей из этих переменных, полагая, что среди них нет равных.

Вариант 3.8

Для произвольных значений *x, a* вычислить

$$y=\begin{cases}\cos\dfrac{x+a}{1-xa} & при\quad ax<1\\ 3{,}5+\cos\dfrac{x+a}{1-xa} & при\quad x>0,\ xa>1\\ ax & в\ остальных\ случаях\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.9

Для произвольного значения x вычислить

$$y=\begin{cases}\dfrac{x}{1+\dfrac{x^2}{1+\dfrac{2x^3}{1+x}}} & при\ x>0\\ 2+x\sin x & при\ x\le 0\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.10

Для произвольных значений x, a вычислить

$$y=\begin{cases}x-2e^{x^2} & при\quad x<0\ и\ a<2\\ x+8\sin 2x & при\quad a>10\\ a^2+\sqrt{|x|} & в\ остальных\ случаях\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.11

Для произвольных значений x, a вычислить

$$y=\begin{cases}x^{a+1}+10a & при\quad x>0\ и\ a>1\\ x\sin 3x & при\quad x>0,\ a<0\\ \cos a^2+x & в\ остальных\ случаях\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.12

Для произвольных значений x, a вычислить

$$y=\begin{cases} x+2a-\sqrt{|a+x|} & \textit{при}\quad x>0\ \textit{и}\ \ a<0 \\ x^2\ln x & \textit{при}\quad x>0,\ a>0 \\ \dfrac{\sin 2x}{(ax+1)^2+1} & \textit{в остальных случаях} \end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.13

Для произвольных значений *x, a* вычислить

$$y=\begin{cases} x+\ln(x^2+1) & \textit{при}\quad a>x \\ x+\sin 3a & \textit{при}\quad a=x \\ a\ln(x^2+1)+e^{x+1} & \textit{при}\quad a<x \end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.14

Ввести с клавиатуры значения переменных *A, B, C*. Вывести на экран идентификатор и значение наименьшей по абсолютной величине переменной. Считать, что равных переменных нет.

Вариант 3.15

Ввести с клавиатуры значения переменных *A, B, C* и найти среди них пару, дающую наибольшую сумму. Вывести на экран идентификаторы искомых переменных и значение суммы.

Вариант 3.16

Для произвольных значений *x, a* вычислить

$$y=\begin{cases} x+2a & \textit{при}\quad x-a>0 \\ x+\dfrac{x+a}{1+\dfrac{2x^2}{1+a^2}} & \textit{при}\quad x-a<-10 \\ 0 & \textit{при}\quad -10\le x-a\le 0 \end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.17

Определить количество чисел кратных 5 среди переменных *K, L, M*. Также вывести на экран идентификаторы переменных удовлетворяющих этому условию.

Вариант 3.18

Для произвольных значений *x, a* вычислить

$$y=\begin{cases}1 & при\ \ a=1\\ \sin 2x & при\ \ a=5\\ \cos x+e^{x} & при\ \ a=6\\ 2x-1 & в\ остальных\ случаях\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.19

Ввести с клавиатуры значения переменных *A, B, C*. Вывести на экран идентификатор и значение наименьшей из этих переменных, полагая, что среди них нет равных.

Вариант 3.20

Для произвольных значений *x, t* вычислить

$$y=\begin{cases}x+t & при\ \ x^{2}+\sin t>0{,}1\\ 2x+t^{2} & при\ \ x^{2}+\sin t<0{,}1\\ x-t & при\ \ x^{2}+\sin t=0{,}1\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.21

Для произвольных значений *x* вычислить

$$y=\begin{cases}5\cos 2x-\sin 8x & при\ \ x>1\\ x+5^{x} & при\ \ x<-2\\ e^{x+3x^{2}} & в\ остальных\ случаях\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.22

Для произвольного значения *x* вычислить

$$y=\begin{cases}((x+1)x+x)x & при\ x>0\\ 0 & при\ x=0\\ x^3+\ln|x| & при\ x<0\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.23

Для произвольного значения *x* вычислить

$$y=\begin{cases}(x+2)3^x & при\ x\ge 2\\ x\sin 4x-6x & при\ x\le 0\\ x\ln 2x-5 & при\ 0<x<2\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.24

Для произвольных значений *x, a* вычислить

$$y=\begin{cases}x+a & при\ x>0\ и\ a\ne 0\\ 2ax+\ln|x| & при\ x\le 0\ и\ a\ne 0\\ x & при\ a=0\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.25

Для произвольного значения *x* вычислить

$$y=\begin{cases}5x+\ln x & при\ x>2\\ \sin x+\cos 2x & при\ 0\le x\le 2\\ \left|\dfrac{\sin x}{2+\cos x}\right| & при\ x<0\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.26

Для произвольных значений *x, a* вычислить

$$y=\begin{cases}2x & при\ a=2\\ 3x+\cos x & при\ a=5\\ \sin 4x+1 & при\ a=9\\ 0 & в\ остальных\ случаях\end{cases}$$

Вывести на экран сообщение, в котором указывается номер выполнившегося условия.

Вариант 3.27

Ввести с клавиатуры значения переменных *A, B, C*. Найти наибольшее из выражений $(AB)^C$, $(AC)^B$, $(BC)^A$, полагая, что среди них нет равных. Вывести на экран сообщение, в котором указывается номер найденного выражения и его значение.

Вариант 3.28

Даны переменные *A, B*. Найти арифметическую операцию (+, *, /), которая дает наибольший результат (считать, что среди них нет равных). Вывести на экран сообщение, в котором указывается операция и ее результат.

Вариант 3.29

Определить количество отрицательных чисел среди переменных *K, L, M*. Также вывести на экран идентификаторы переменных удовлетворяющих этому условию.

Вариант 3.30

Определить, есть ли среди переменных *P, R, S* положительные числа.

Лабораторная работа № 4

Программирование процессов циклической структуры

Цель работы: изучение основных приемов проектирования и отладки программ циклической структуры на Паскале.

Процессы циклической структуры

В **циклических алгоритмах** выполнение некоторых операторов (групп операторов) осуществляется многократно с одними и теми же или модифицированными данными.

Циклический алгоритм должен содержать следующие основные блоки:

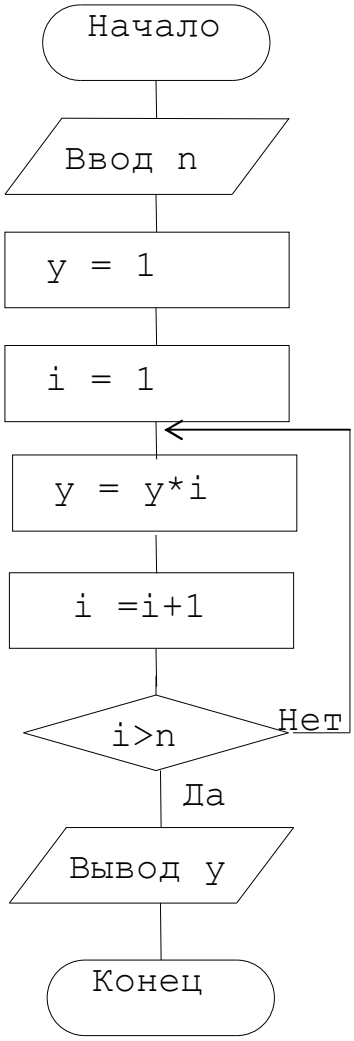
- определение начальных значений параметров (инициализация);
- проверка условия продолжения или окончания вычислений;
- вычисление операций при заданных параметрах цикла (тело цикла);
- изменение значений параметров цикла.

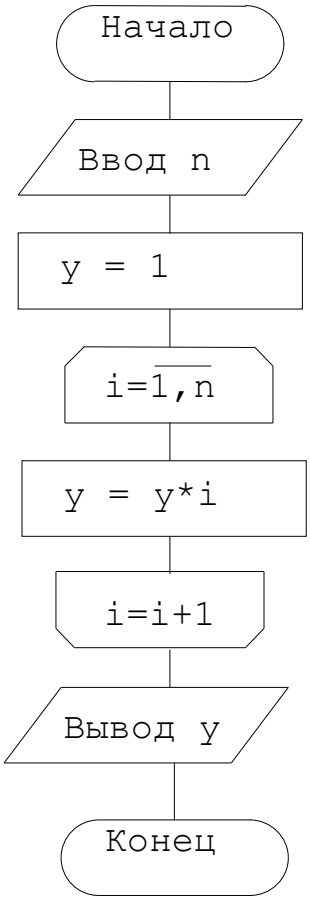
Для реализации циклических структур в схемах алгоритма используется символ "границы цикла", но в некоторых случаях удобнее использовать символ "решение".

Символ **граница цикла** состоит из двух частей и отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения цикла помещаются внутри символа в начале или конце в зависимости от расположения операции, проверяющей условие завершения цикла.

На рисунке 4.1 приведены схемы алгоритма вычисления факториала (y=n!), использующие символ "решение" (рисунок 4.1,а) и символ "границы цикла" (рисунок 4.1,б).

Циклический алгоритм, представленный на рисунке 4.1,а, содержит все необходимые блоки: определение начальных значений параметров (i = 1); вычисление операций при заданных параметрах цикла (y = y*i); изменение значений параметров цикла (i =i+1); проверка условия продолжения или окончания вычислений (i>n).





а) б)

Рисунок 4.1

Операторы цикла языка Паскаль

В Паскале есть три оператора цикла.

Оператор for определяет цикл с управляющей переменной (параметром) и обеспечивает выполнение оператора в теле цикла заданное число раз. В операторе указывается переменная любого перечислимого типа (управляющая переменная) и определяется последовательность значений, которые должна пробежать эта переменная в ходе выполнения цикла.

Формат

for <переменная>:=<выражение1> to <выражение2> do <оператор>;

или

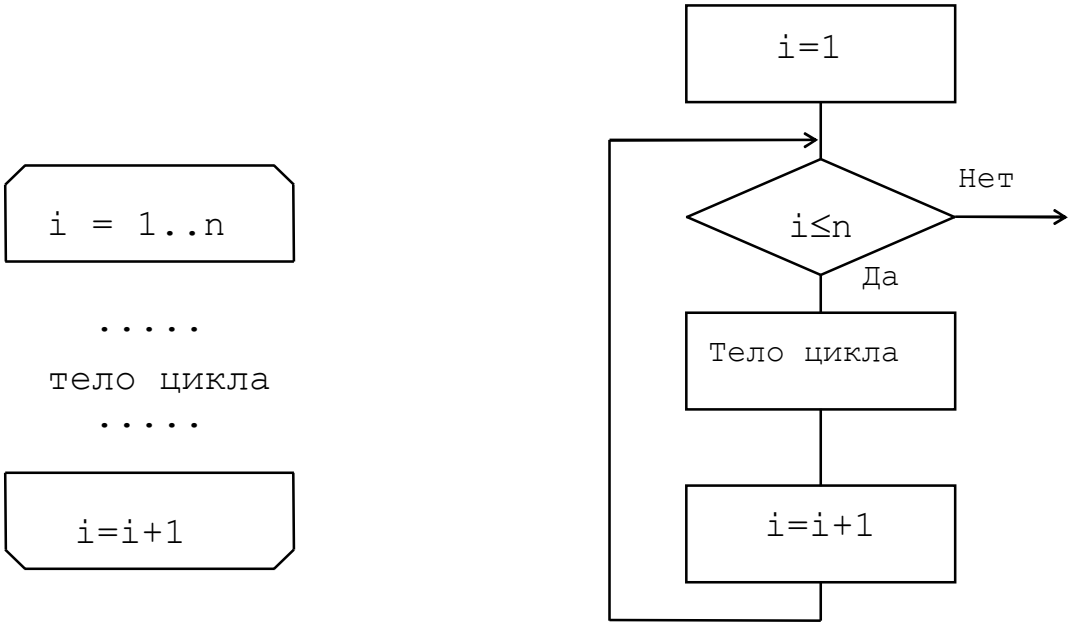
for <переменная>:=<выражение1> downto <выражение2> do <оператор>;

<Выражение1> и <выражение2> определяют, соответственно, начальное и конечное значение управляющей переменной и должно быть того же типа, что и управляющая переменная. Изменение значений управляющей переменной осуществляется в порядке возрастания (to) или убывания (downto) их порядковых номеров в соответствующем типе данных.

Пример: for i:=2 to 10 do x:= x+i;

Управляющая переменная должна быть перечислимого типа. Из стандартных типов - это целый, символьный и логический типы. Шаг цикла for всегда постоянный и равен интервалу между двумя ближайшими значениями типа параметра цикла (для целого типа это 1). Изменение параметра цикла и контроль за выходом из цикла осуществляется автоматически.

Варианты схем алгоритма для реализации циклических структур с помощью оператора for (при изменении параметра от 1 до n) приведены на рисунке 4.2.



а) б)

Рисунок 4.2

Рассмотрим порядок выполнения действий в операторе

for p:=kn to kv do <оператор>;

Выполнение оператора for начинается с присвоения параметру цикла начального значения (p:=kn). Затем производится проверка условия p <= kv. Если условие истинно, то выполняется оператор в цикле, а затем пара-

метру цикла присваивается следующее значение и вновь производится проверка условия p <= kv. Если условие p <= kv не справедливо, то <оператор> в цикле не выполняется, а управление передается оператору, следующему за оператором цикла. Если используется параметр целого типа, то это означает его увеличение на единицу при каждом новом выполнении расположенного в цикле оператора (задать шаг отличный от 1 или -1 в этом случае нельзя!).

Не рекомендуется изменять параметр цикла внутри цикла.

Войти в цикл можно только через его начало, а выйти либо при исчерпании значений параметра цикла, либо по метке, расположенной вне данного цикла.

Следует помнить, что при выходе из цикла по исчерпании значений параметра цикла значение этого параметра не определено. Хотя обычно параметр остается равным конечному значению, не следует опираться на это в своих программах. Исключение - выход из цикла переходом по goto. В этом случае значение переменной (параметра цикла) остается таким же, каким было на момент выполнения оператора goto.

Если в цикле должно быть выполнено несколько операторов, то следует использовать составной оператор.

Ниже приведена программа на языке Паскаль, реализующая вычисление факториала y=n! (рисунок 1).

```
Program P401;
var
        n, i, y: integer;
begin
        readln(n);
        y:=1;
        for i:=1 to n do
                y:= y*i;
        writeln('y=', y);
end.
```

Оператор while определяет цикл с предварительной проверкой условия.

Варианты схем алгоритма для реализации циклических структур с помощью оператора while приведены на рисунке 4.3.

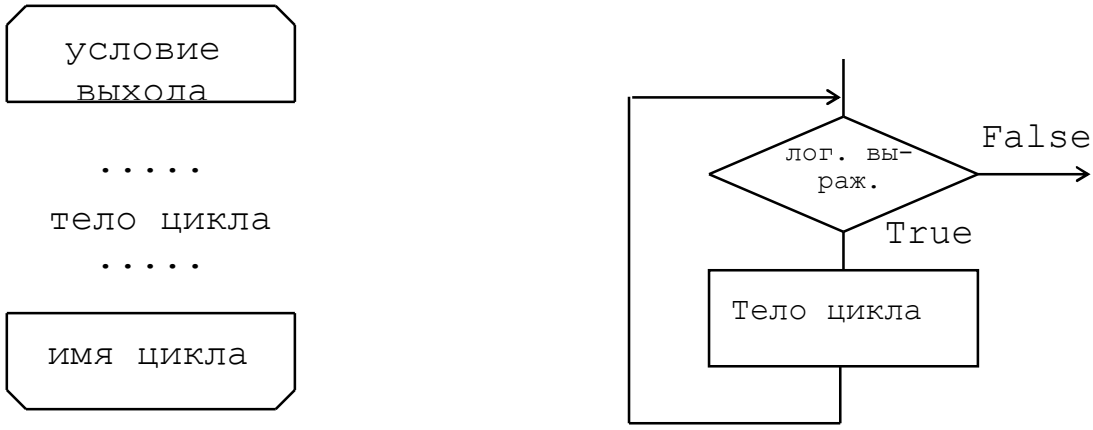


Рисунок 4.3

Формат оператора:

while <логическое выражение> do <оператор>;

Выполнение оператора while начинается с проверки логического выражения. Если оно имеет значение true, то <оператор> будет выполнен и произойдет переход к проверке логического выражения. Если логическое выражение имеет значение false, то выполнение цикла будет завершено.

Изменение операндов в логическом выражении для обеспечения выхода из цикла должно быть предусмотрено в самом цикле.

Пример. Подсчитать количество нулей, которыми заканчивается число n.

```
Program P402;
var
        i, d, n: integer;
begin
        write ('Введите натуральное число: '); readln(n);
        i:=0; d:=10;
        while n mod d=0 do
                begin
                i:=i+1;
                d:=d*10
                end;
        writeln('Число нулей: ', i);
end.
```

Если логическое выражение сразу же равно False - оператор в цикле не выполнится ни разу.

Если в цикле необходимо выполнить несколько операторов, то необходимо использовать составной оператор.

Оператор repeat определяет цикл с последующей проверкой условия.

Формат:

repeat

<список операторов>

until <логическое выражение>;

<Список операторов> может содержать больше одного оператора. Указанные операторы будут выполняться в цикле до тех пор, пока <логическое выражение> не примет значение True.

Изменение операндов в логическом выражении для обеспечения выхода из цикла должно быть предусмотрено в самом цикле.

Варианты схем алгоритма для реализации циклических структур с помощью оператора repeat приведены на рисунке 4.4.

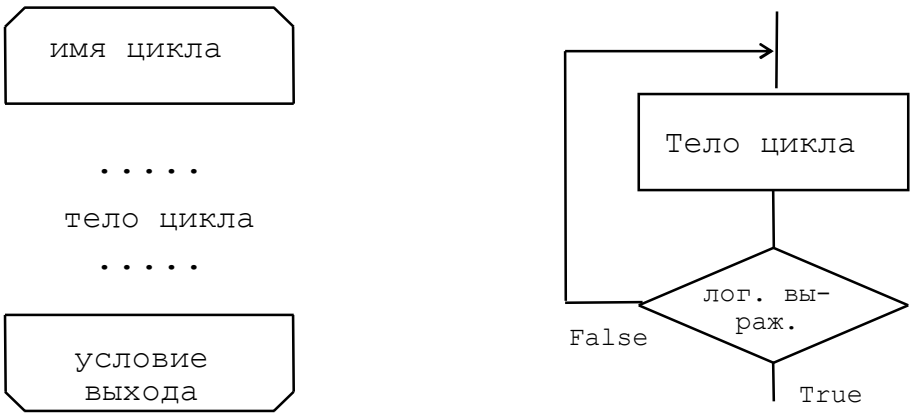


Рисунок 4.4

Пример. Вычислить сумму последовательности чисел, введенных с клавиатуры. Признак окончания ввода - равенство введенного числа нулю.

```
Program P403;
var
    s,a:integer;
begin
    s:=0;
    repeat
        readln(a);
        s:=s+a;
```

```
    until a=0;
    writeln('Сумма: ',s:6);
end.
```

Тело цикла с постусловием всегда будет выполнен хотя бы один раз.

Входить в цикл можно только через заголовок цикла!

Из циклов while и repeat можно выйти не только по выполнении условия, но и с помощью оператора goto.

В циклах repeat, while и for можно использовать две стандартные процедуры - Break и Continue. Процедура Break позволяет досрочно выйти из цикла, не дожидаясь выполнения условия выхода. Процедура Continue позволяет начать новую итерацию цикла, даже если предыдущая не завершена.

Если телом цикла является циклическая структура, то такие циклы называют вложенными или сложными. Цикл, содержащий в себе другой цикл, называют внешним. Цикл, содержащийся в теле другого цикла, называют внутренним.

Внутренний и внешний циклы могут быть любыми из трех рассмотренных видов: циклами с параметром, циклами с предусловием, циклами с постусловием. Правила организации как внешнего, так и внутреннего циклов такие же, как и для простого цикла каждого из этих видов. Однако при построении вложенных циклов необходимо соблюдать следующее дополнительное условие: все операторы внутреннего цикла должны полностью лежать в теле внешнего цикла.

Параметры циклов разных уровней изменяются не одновременно. Вначале все свои значения изменит параметр цикла наивысшего уровня вложенности при фиксированных (начальных) значениях параметров циклов с меньшим уровнем. Затем изменяется на один шаг значение параметра цикла следующего уровня и снова полностью выполняется самый внутренний цикл и т.д. до тех пор, пока параметры циклов всех уровней не примут все требуемые значения.

Нельзя в двух циклах использовать один и тот же идентификатор в качестве параметра.

Итерационные процессы

Типичный пример использования операторов цикла – реализация итерационных процессов.

Например, требуется найти сумму ряда

$$y=\sum_{n=0}^{\infty}\frac{x^n}{n!} \qquad (-\infty<x<+\infty)$$

с точностью ε=0.001.

Алгоритм вычисления y сводится поочередному вычислению элементов ряда с включением их в сумму до тех пор, пока не будет достигнута требуемая точность.

При определении суммы членов ряда можно использовать рекуррентную формулу для получения следующего члена ряда.

Сумма ряда может быть записана и в таком виде

$$y = \frac{x^0}{1} + \frac{x^1}{1} + \frac{x^2}{1 \cdot 2} + \frac{x^3}{1 \cdot 2 \cdot 3} + \ldots$$

Тогда, обозначив *n*-е слагаемое как a_n, можно записать

$$y = a_0 + a_1 + a_2 + a_3 + \ldots a_n + \ldots.$$

Отметим, что

$$a_0 = \frac{1}{1}; a_1 = \frac{x}{1}; a_2 = \frac{x}{1} \cdot \frac{x}{2} = a_1 \cdot \frac{x}{2}; a_3 = \frac{x^2}{1 \cdot 2} \cdot \frac{x}{3} = a_2 \cdot \frac{x}{3}.$$

Для получения рекуррентной формулы вычислим отношение текущего элемента ряда к предыдущему:

$$\frac{a_n}{a_{n-1}} = \frac{\frac{x^n}{n!}}{\frac{x^{n-1}}{(n-1)!}} = \frac{x^n (n-1)!}{x^{n-1} n!} = \frac{x(n-1)!}{(n-1)! n} = \frac{x}{n}$$

откуда

$$a_n = a_{n-1} \frac{x}{n}.$$

Т.о. получили следующий алгоритм вычисления суммы:

а) вычисление слагаемого по рекуррентной формуле $a_n = a_{n-1} \frac{x}{n}$;

б) включение его в сумму;

в) переход к следующему слагаемому.

Когда следует закончить суммирование?

Точность вычисления задана - ε, а рассматриваемые в данном пособии ряды таковы, что с ростом *n* значение слагаемого (по абсолютной величине) уменьшается. Поэтому вычисление суммы завершается, если оказывается что $a_n < \varepsilon$.

Для того чтобы начать рекуррентный процесс необходимо задать начальные значения для суммы и для слагаемого, вычисляемого по рекуррентной формуле.

Начальное значение суммы очевидно равно 0.

Учитывая, что 0!=1, можно вычислить нулевой элемент ряда и сразу его включить в сумму, а рекуррентный процесс начинать с первого.

Тогда начальное значение слагаемого следует взять равным 1 ($\frac{x^0}{0!}$).

Количество слагаемых в сумме будет зависеть от заданной точности.

Словесное описание алгоритм вычисления сумм ряда может выглядеть следующим образом:

1. Задать начальные значения.
2. Увеличить счетчик слагаемых.
3. Вычислить очередное слагаемое.
4. Включить слагаемое в сумму.
5. Если точность не достигнута, перейти к п. 2. Иначе к п. 6.
6. Вывод результата.

Порядок выполнения работы

1. Изучить приемы проектирования программ циклической структуры.
2. Изучить операторы цикла языка Паскаль.
3. Изучить приемы реализация итерационных процессов.
4. Вывести рекуррентную формулу (в соответствии с вариантом).
5. Разработать алгоритм и программу на языке Паскаль для вычисления суммы ряда (в соответствии с вариантом).
6. Подготовить отчет.

Варианты заданий

Вариант 4.1

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{(2n+1)!} \qquad (-\infty<x<+\infty)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=\sin x$.

Вариант 4.2

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!} \qquad (-\infty<x<+\infty)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=\cos x$.

Вариант 4.3

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{(2n+1)!} \qquad (-\infty<x<+\infty)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=\mathrm{sh}\,x = =\frac{e^{x}-e^{-x}}{2}$.

Вариант 4.4

Используя рекуррентную формулу, вычислить сумму ряда

$$y=-\sum_{n=1}^{\infty}\frac{x^{n}}{n} \qquad (-1<x<+1)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=ln(1-x)$.

Вариант 4.5

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1} \qquad (-1<x<+1)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $Arth\,x=\frac{1}{2}\ln\frac{1+x}{1-x}$.

Вариант 4.6

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{(x-1)^{n}}{n} \qquad (0<x\le+2)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=ln(x)$.

Вариант 4.7

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^{n}}{n} \qquad (-1<x\le+1)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=ln(x+1)$.

Вариант 4.8

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\frac{(x-1)^n}{nx^n} \qquad (x>\frac{1}{2})$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=ln(x)$.

Вариант 4.9

Используя рекуррентную формулу, вычислить сумму ряда

$$y=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1} \qquad (-1<x<+1)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=\ln\frac{1+x}{1-x}$.

Вариант 4.10

Используя рекуррентную формулу, вычислить сумму ряда

$$y=1+\sum_{n=1}^{\infty}\frac{(x\ln a)^n}{n!} \qquad (-\infty<x<+\infty)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=a^x$.

Вариант 4.11

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\left(\frac{2}{n!}+\frac{1}{x^n}\right) \qquad (-\infty<x<+\infty)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.12

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\frac{\pi}{2}-\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{2n+1} \qquad (-1<x<+1)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y = arcctg\, x = \frac{\pi}{2} - arctg\, x$.

Вариант 4.13

Используя рекуррентную формулу, вычислить сумму ряда

$$y=1+\sum_{n=1}^{\infty}\frac{x^{2n}}{(2n)!} \qquad (-\infty<x<+\infty)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $y=\mathrm{sh}\, x = =\frac{e^{x}+e^{-x}}{2}$.

Вариант 4.14

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}} \qquad (|x|>1)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Примечание. Сумма ряда сходится к значению $Arth\, x = \frac{1}{2}\ln\frac{x+1}{x-1}$.

Вариант 4.15

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^{n-1}}{10^{n}} \qquad (-10<x<+10)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.16

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\left(\frac{1}{2^{n}}+\frac{1}{3^{n}}\right)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.17

Используя рекуррентную формулу, вычислить сумму ряда

$$y = \sum_{n=1}^{\infty} \frac{2n-1}{2^n}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.18

Используя рекуррентную формулу, вычислить сумму ряда

$$y = \sum_{n=1}^{\infty} \frac{1}{2^n (3n-2)}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.19

Используя рекуррентную формулу, вычислить сумму ряда

$$y = \sum_{n=1}^{\infty} \frac{10^n}{n!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.20

Используя рекуррентную формулу, вычислить сумму ряда

$$y = \sum_{n=1}^{\infty} \frac{n!}{(2n)!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.21

Используя рекуррентную формулу, вычислить сумму ряда

$$y = \sum_{n=1}^{\infty} \left(\frac{1}{10^n} + \frac{1}{n!} \right)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.22

Используя рекуррентную формулу, вычислить сумму ряда

$$y = \sum_{n=1}^{\infty} \frac{3^n}{10^n n!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.23

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\frac{3^n n!}{(2n)!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.24

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\frac{(2n)!}{300^n}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.25

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\frac{n!}{(2n+1)!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.26

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\frac{10^n}{(2n)!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.27

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}(-1)^n\frac{2^{2n-1}}{(2n)!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.28

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^{2n}}{2^{2n+1}} \quad (x<2)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.29

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^{2n+1}}{(2n)!}$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Вариант 4.30

Используя рекуррентную формулу, вычислить сумму ряда

$$y=\sum_{n=1}^{\infty}\frac{2^{n}}{x^{n+1}} \quad (x>2)$$

с точностью 1) ε = 0.01; 2) ε = 0.001. Определить количество элементов ряда, включенных в сумму.

Лабораторная работа № 5

Массивы

Цель работы: изучение основных приемов работы с массивами в Паскале.

Структура данных представляет собой группу связанных друг с другом данных, размещенных в памяти компьютера. С этими сгруппированными данными можно работать по отдельности, но есть операции, которые применимы ко всей структуре как единому целому.

Переменная, имеющая структуру **массива** - это совокупность элементов одного и того же типа. Для массива характерно следующее:

1) каждый элемент массива может быть явно обозначен и к нему имеется прямой доступ;

2) число элементов массива определяется при его описании и в дальнейшем не меняется.

Для обозначения элементов массива используется имя переменной массива и так называемые индексы, которые однозначно указывают нужный элемент. Так как индексы могут быть вычисляемыми объектами, они должны иметь один из допустимых типов данных, который называется типом индекса массива. Тип индекса может быть только перечислимым типом, причем тип integer не допускается. Однако допускается использование типа диапазон целого.

Массивы описываются в разделе описания переменных, причем все элементы массива должны быть одного типа, который называется *базовым типом массива*.

Формат

var <идентификатор>: array[<тип индекса>] of [<базовый тип>];

Примеры описаний:

```
var x: array [1..10] of real;
var y: array ['a'..'z'] of integer;
```

Элемент массива имеет те же свойства, что и переменная того же типа и, следовательно, может входить в состав выражений.

Чтобы использовать элемент массива, необходимо указать имя массива, а за ним в квадратных скобках индекс. Индекс может быть как константой, так и выражением.

Пример:

```
x[2]
x[k+2]
y['b'].
```

Тип индекса обязательно должен совпадать с типом индекса, указанным при описании массива, и иметь значение, лежащее в указанном диапазоне.

Массивы могут иметь более одного измерения. Например, чтобы описать двумерный массив необходимо записать:

var m: array[1..10, 1..50] of real;

и тогда элемент массива

m[8,9] или m[8][9].

Двумерные массивы часто рассматривают как матрицы и тогда говорят, что m[8,9] - это элемент матрицы из восьмой строки и девятого столбца.

Соответственно может быть n-мерный массив, тогда его элементы определяются указанием n индексных выражений.

Кроме приведенного выше, существует еще один прием описания массива. Базовым типом массива может быть другой массив.

Пример.

type xtype = array[1..50] of real;

var m: array[1..10] of xtype;

Массив m - двумерный массив, имеющий 10 строк и 50 столбцов.

Значение одного массива может быть присвоено другому массиву эквивалентного типа в одном операторе присваивания:

```
var x, z: xtype;
....
                x:=z;
```

Все элементы массива x получат значения равные соответствующим значениям массива z.

Пример.

В одномерном массиве заменить нулями все элементы, значения которых больше значения последнего элемента.

```
Program P501;
const
    n=20;
var
    x:array [1..n] of real;
    s:real;
    i:integer;
begin
    for i:=1 to n do readln (x[i]);
    for i:=1 to n do
        if x[i]>x[n] then x[i]:=0;
```

```
    for i:=1 to n do writeln(x[i]);
end.
```

Пример.

Определить количество строк в квадратной матрице, имеющих четное число положительных элементов.

```
Program P501;
type
    astr = array [1..5] of real;
    atype = array [1..5] of astr;
var
    a: atype;
    n,    {количество строк в матрице}
    m,    {количество строк в матрице, имеющих четное число положительных элементов}
    k,    {количество положительных элементов в строке}
    i, j: integer;
begin
    write('n='); readln(n);
    for i:=1 to n do
        for j:=1 to n do
            begin
            write('a[',i,',',j,']=');
            readln(a[i,j]);
            end;
    m:=0;
    for i:=1 to n do
        begin
        k:=0;
        for j:=1 to n do
            if a[i,j] > 0 then k:=k+1;
        if k mod 2 = 0 then m:=m+1;
        end;
    writeln ('Количество строк, имеющих четное число положительных элементов, - ',m);
end.
```

Порядок выполнения работы

1. Изучить приемы работы с массивами в языке Паскаль.

2. Разработать алгоритм и программу на языке Паскаль (в соответствии с вариантом), с обязательным использованием массива.

3. Подготовить отчет.

Варианты заданий

Вариант 5.1.

Определить номера столбцов в матрице, имеющих четное число положительных элементов.

Вариант 5.2.

Дана целочисленная матрица. Есть ли в ней одинаковые строки?

Вариант 5.3.

Найти сумму элементов матрицы (NxN), расположенных в строках с отрицательными элементами на главной диагонали.

Вариант 5.4.

Дан одномерный массив целых чисел. Переставить элементы в массиве таким образом, чтобы сначала следовали все отрицательные числа, а затем все остальные.

Примечания. Не использовать другой массив.
Сохранить порядок следования элементов.

Вариант 5.5.

В матрице определить номер и значение максимального отрицательного элемента.

Вариант 5.6.

В массиве из 40 элементов подсчитать число элементов с четными номерами, имеющими значение FALSE.

Вариант 5.7.

В произвольной последовательности чисел, заданной в виде одномерного массива, найти самую длинную последовательность, упорядоченную по возрастанию.

Вариант 5.8.

Дан числовой массив. Найти отрезок массива максимальной длины, в котором первое число равно последнему. Пример: для исходных данных 1 2 3 5 8 3 2 6 ответ 2 3 5 8 3 2.

Вариант 5.9.

Из символьного массива исключить все элементы, содержащие символ 'A'.

Вариант 5.10.

Составить программу, в которой меняются местами каждый элемент с нечетным индексом и следующий за ним элемент с четным индексом, входящие в одномерный массив чисел.

Вариант 5.11.

В двумерном массиве определить номера строк, содержащих только положительные элементы.

Вариант 5.12.

В матрице заменить все строки с наибольшей суммой элементов последней строкой.

Вариант 5.13.

Составить программу для формирования вектора $X(x_1, x_2, ...,x_n)$, где x_1 совпадает с первой ненулевой компонентой вектора $Y(y_1, y_2,..., y_m)$ при просмотре его компонент слева направо, x_2 - со следующей ненулевой компонентой и т.д.

Вариант 5.14.

Транспонировать матрицу, не используя дополнительный массив.

Вариант 5.15.

Даны два символьных массива, в каждом из которых нет повторяющихся элементов. Есть ли в массивах разные элементы?

Вариант 5.16.

Сформировать массив чисел, состоящий из целых положительных степеней числа 2, не превышающих некоторое число N.

Вариант 5.17.

Дан символьный массив. Вывести на экран наиболее часто встречающийся элемент. Если таких элементов несколько, то вывести все.

Вариант 5.18.

Определить, каких латинских букв нет в некотором символьном массиве. Подсчитать количество этих букв.

Вариант 5.19.

Элементы массива циклически сдвинуть на **k** позиций влево. Пример:

123456

после сдвига на две позиции 345612

Вариант 5.20.

Определить сколько различных чисел являются элементами некоторого целочисленного массива и найти их сумму.

Вариант 5.21.

Даны матрица A размером MxM и вектор B, состоящий из M элементов. Вектор B умножить на 20, если наибольший элемент матрицы A (в предположении, что такой элемент единственный) находится на главной диагонали, и на 0,5 - в противном случае.

Вариант 5.22.

Элементы каждого из двух массивов упорядочены по неубыванию. Объединить элементы этих массивов в один массив так, чтобы они снова оказались упорядоченными по неубыванию.

Вариант 5.23.

Дан массив из четырех элементов. Значение каждого элемента – цифра. Составить наибольшее число из этих цифр (каждую цифру использовать один раз).

Вариант 5.24.

Дана последовательность чисел. Найти сумму чисел этой последовательности, расположенных между максимальным отрицательным и минимальным положительным числами. Считать, что такие числа в последовательности обязательно есть.

Вариант 5.25.

Даны две последовательности чисел. Найти наименьшее среди тех чисел последовательности, которые не входят во вторую последовательность. (Считать, что хотя бы одно такое число есть.)

Вариант 5.26.

В двумерном массиве найти номера строк, имеющих и положительные и отрицательные элементы. Определить номер строки с наибольшим количеством отрицательных элементов (если таких строк несколько, то вывести номера всех этих строк).

Вариант 5.27.

Определить номер столбца в матрице, сумма положительных элементов которой является наибольшей. Если таких столбцов несколько, то вывести номера всех этих столбцов.

Вариант 5.28.

Дан массив. Элементами массива являются строчные латинские буквы. Вывести на экран дисплея в алфавитном порядке все буквы, которые входят в массив по одному разу.

Вариант 5.29.

Даны матрица A. Получить матрицу B из матрицы A удалением n-ой строки и k-го столбца.

Вариант 5.30.

Определить, является ли заданная квадратная матрица симметричной относительно главной диагонали.

Лабораторная работа № 6

Функции и процедуры

Цель работы: изучение основных приемов работы с подпрограммами в Паскале.

Рассмотрим использование в Паскале структур, которые в программировании обычно называют подпрограммами. **Подпрограмма** - это часть программы, оформленная в виде отдельной синтаксической конструкции и снабженная именем. Подпрограммы, как правило, реализуют некоторый завершенный алгоритм. В Паскале два вида подпрограмм: *процедуры* и *функции*. Процедуры и функции позволяют включать в основной программный блок дополнительные блоки. В Паскале процедуры и функции описываются в соответствующих разделах описательной части, причем для каждой подпрограммы отводится свой раздел.

Чаще всего подпрограммы используют для того, чтобы однажды написанный фрагмент программы можно было многократно использовать для решения сходных задач.

Использование подпрограмм также позволяет повысить скорость разработки и отладки программы, увеличить ее надежность.

Рассмотрим на примере использование подпрограмм в Паскале.

Пример. Вычислить число сочетаний из n по m

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

Очевидно, что многократное вычисление факториала будет выполняться каждый раз с использованием одного и того же алгоритма, одних и тех же программных средств. Чтобы избежать этого, и, тем самым, упростить и сократить программу воспользуемся функцией.

Схема алгоритма приведена на рисунке 6.1. Отметим, что для каждого блока разрабатывается отдельная схема алгоритма, хотя обращение из одной схемы к другой допускается.

Ниже приведена программа на языке Паскаль, реализующая вычисление числа сочетаний из n по m (рисунок 6.1).

```
Program P601;
var
      n, m, c: integer;
      n1, m1, nm: integer;
function Fact(n:integer):integer;
      var
            i, p: integer;
```

```
        begin
        p:=1;
        for i:=1 to n do p:=p*i;
        Fact:=p;
        end;

begin
        write('n='); readln(n);
        write('m='); readln(m);
        n1:=Fact(n);
        m1:=Fact(m);
        nm:=Fact(n-m);
        c:=n1 div m1 div nm;
        writeln('c=',c);
end.
```

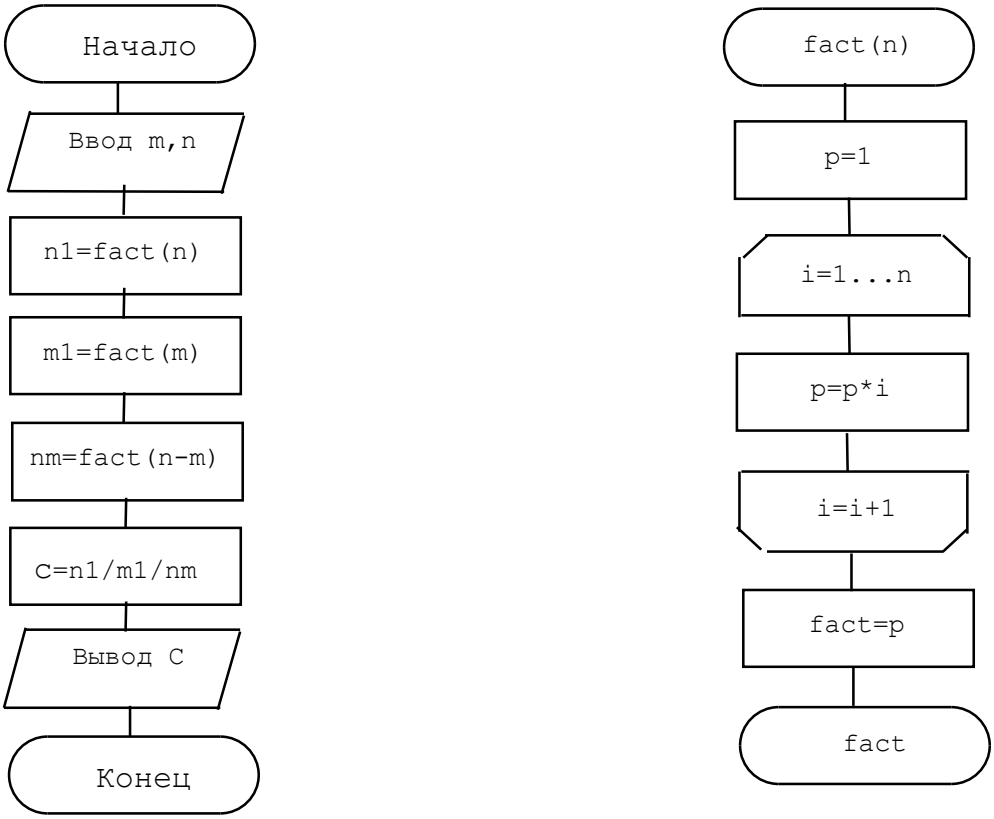


Рисунок 6.1

Вызов подпрограммы, то есть выполнение действий, заданных в подпрограмме в виде операторов, может быть произведен в некоторой точке программы посредством указания имени этой подпрограммы. При этом управление передается первому оператору в подпрограмме. После выполнения всех операторов подпрограммы управление передается в точку вызова.

Подпрограмма может быть разработана для различных случаев применения. Иными словами, перед выполнением подпрограммы можно передать ей некоторую информацию из точки вызова, настроив ее соответствующим образом. Такая настройка подпрограмм реализуется с помощью механизма *параметров* и способствует большей гибкости и универсальности подпрограмм.

Функции вычисляют единственное значение простого типа. Описание функции состоит из двух частей: заголовка функции и тела функции.

В заголовке функции указывается имя функции, описывается список формальных параметров (если они присутствуют) и указывается тип функции (тип результата функции).

Тело функции имеет точно такую же структуру, как программа: в нем могут присутствовать все разделы и на него распространяются общие правила локализации.

function <идентификатор>(<список формальных параметров>)**:**<тип функции>;

Итак, описание функции начинается с зарезервированного слова function, за которым следует имя этой функции и, возможно, список формальных параметров. После списка параметров ставится двоеточие и указывается тип результата функции. В секции формальных параметров перечисляются идентификаторы формальных параметров, причем для каждого идентификатора определяется его тип.

Все формальные параметры, а также метки, константы, типы, переменные, процедуры и функции, описанные в блоке, представляющем тело функции, *локальны* по отношению к описанию функции, которое и является областью действия этих объектов. *Они не существуют вне описания функции.* Их значения становятся неопределенными при выходе из описания функции.

Описание функции выполняет следующие функции:

1) идентификатор, использованный после зарезервированного слова function, определяется как имя функции. Его не нужно определять где-либо в другом месте программы. Область его действия - блок, непосредственно содержащий данное описание функции. Следовательно, внутри блока, содержащего описание функции, идентификатор функции должен отличаться от всех других идентификаторов.

2) идентификаторы, использованные в качестве формальных параметров, можно использовать в блоке, представляющем собой тело функции, без каких-либо дополнительных описаний. Они должны отличаться от всех других идентификаторов, использованных в теле процедуры, и друг от друга.

3) блок функции связывается с идентификатором этой функции.

Таким образом, при описании функции определяются все объекты, с которыми они работают. Если в теле функции использованы некоторые идентификаторы, не описанные в теле функции и не являющиеся формальными параметрами, то они считаются глобальными по отношению к данному описанию функции. Их описания или определения можно найти в начале одного из блоков, охватывающих данное описание функции.

В теле функции обязательно должен присутствовать оператор присваивания, в левой части которого стоит идентификатор этой функции. Именно этот оператор присваивания и определяет значение, вычисленное с помощью этой функции. Если такой оператор присваивания отсутствует, то значение, вычисленное с помощью функции, не определено. Таких операторов присваивания в теле функции может быть несколько. Очевидно, что значение, которое вычисляется функцией, задается последним выполненным до завершения выполнения функции оператором.

Обращение к функции осуществляется с помощью указателя функции, состоящего из идентификатора вызываемой функции и списка *фактических* параметров, заключенного в круглые скобки. Указатели функции должны использоваться в выражениях.

Пример: x:=a*fun(a,b,c);
y:=fun(a,b,c);

Между формальными и фактическими параметрами устанавливается взаимно однозначное соответствие, путем пересчета их слева на право: первому слева формальному параметру ставиться в соответствии первый слева фактический параметр, второму формальному - второй фактический и т. д. Число формальных параметров должно в точности равняться числу фактических параметров. Следовательно, если в описании функции отсутствует список формальных параметров, то и при вызове функции должен отсутствовать список фактических параметров. Типы фактических параметров должны соответствовать типам формальных параметров.

В качестве фактических параметров функции могут выступать константы, переменные и выражения соответствующих типов. В качестве формальных параметров функции могут выступать только переменные!

Выполнение обращения к функции состоит в следующем: все формальные параметры заменяются соответственно фактическими. После этого создается динамический экземпляр блока функции, который и выполня-

ется.

Описания процедуры позволяет связать идентификатор с процедурным блоком. Процедуру можно затем активизировать с помощью оператора процедуры.

Описание процедуры состоит из двух частей: заголовка процедуры и тела процедуры.

В заголовке процедуры указывается имя процедуры и описывается список формальных параметров (если они присутствуют).

Все раннее сказанное о функции справедливо и для процедуры. Вместе с тем имеются и некоторые отличия. Одно (чисто внешнее): заголовок процедуры начинается со слова procedure. Другие более существенны, а именно:

а) процедура не имеет типа;

б) процедура в качестве результата может иметь несколько значений, массивы и т.д.

Тело процедуры имеет точно такую же структуру, как программа: в нем могут присутствовать все разделы и на него распространяются общие правила локализации.

Формат:

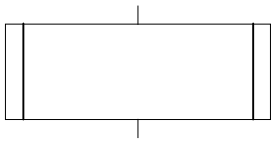
procedure <идентификатор> (<список формальных параметров>);

Итак, описание процедуры начинается с зарезервированного слова procedure, за которым следует имя этой процедуры и, возможно, список формальных параметров. В секции формальных параметров перечисляются идентификаторы формальных параметров, причем для каждого идентификатора определяется его тип.

Для обращения к описанию процедуры используется **оператор процедуры.**

Формат:

<идентификатор процедуры>(<список фактических параметров>);



В схеме алгоритма оператору процедуры соответствует символ "предопределенный процесс", который отображает использование готовых алгоритмов или программ;

Оператор процедуры состоит из идентификатора процедуры (той, к которой необходимо обратиться) и, возможно, списка фактических параметров. Фактические параметры отделяются друг от друга запятыми, а весь список заключается в круглые скобки.

Различают *пять видов формальных параметров*: параметры-значения; параметры-переменные; параметры-процедуры; параметры-функции. В последних двух случаях фактическими параметрами являются идентификаторы процедур и идентификаторы функций соответственно.

Рассмотрим характер использования различных параметров. Если в секции формальных параметров списку идентификаторов предшествует зарезервированное слово var, то все параметры из этого списка являются параметрами-переменными. В качестве фактического параметра, соответствующего формальному параметру-переменной, должна использоваться переменная. Тип фактического параметра должен быть эквивалентным типу, заданному в спецификации параметра-переменной.

При обращении к описанию процедуры с помощью оператора процедуры все действия, задаваемые телом процедуры, выполняются над фактическим параметром. Если переменная, в процессе выполнения процедуры получит новое значение, она сохранит его и после того как процедура будет выполнена.

Пример: procedure A(var a:real; b:integer);

Если в секции формальных параметров никакой символ не предшествует списку параметров, то все параметры из этого списка являются параметрами-значениями. Фактическим параметром, соответствующим параметру-значению, должно быть выражение (в частности - идентификатор переменной или константы). Формальный параметр в этом случае - просто локальная переменная вызываемой процедуры. В качестве начального значения для формального параметра пересылается текущее значение соответствующего фактического параметра. Значение формального параметра может изменяться при выполнении процедуры, однако никакого влияния на фактические параметры это не оказывает. Следовательно, параметр-значение не может представлять собой результат работы процедуры.

Фактический параметр, соответствующий формальному параметру-процедуре, должен быть идентификатором процедуры, определение которого содержится в блоке программы. Аналогично, параметру-функции должен соответствовать идентификатор функции.

Использовать параметры-константы рекомендуется при передаче данных большого объема с гарантией сохранения их значений.

Работа с формальными параметрами-константами внутри подпрограммы ведется как с обычными локальными константами. Только эти константы принимают значения выражений, которые находятся в фактических параметрах. Им не выделяется новая память как локальным переменным. Запрещается изменять их значения во время выполнения подпрограммы.

Заголовок процедуры с параметрами-константами имеет вид:

Procedure Proc(const p1: type1; p2: type2);

Если в качестве параметров выступают *массивы*, то нельзя записать такой заголовок процедуры:

procedure aa(a:array[1..10] of real);

так как согласно правилам языка типы формальных и фактических параметров должны быть эквивалентны. Поэтому, прежде чем записать заголовок процедура, необходимо задать следующие определения:

type massiv=array[1..10] of real;

и тогда

procedure aa(a:massiv);

Отметим, что индексированные переменные можно использовать в качестве фактических параметров, но **нельзя** использовать в качестве формальных.

Пример.

Найти суммы элементов строк матрицы.

```
Program P602;
type
        xt = array [1..10] of real;
        at = array [1..10] of xt;
var
        a:at;
        i, j, n: integer;
procedure sum(x:xt; n:integer);
        var
                s: real;
                j: integer;
        begin
                s:=0;
                for j:=1 to n do s:=s+x[j];
                writeln(s);
        end;
begin
        readln(n);
        for i:=1 to n do
                for j:=1 to n do
                        readln(a[i,j]);
        for i:=1 to n do sum(a[i], n);
end.
```

Отметим, что передача в подпрограмму части массива возможна не всегда. Чаще передается массив полностью. Тогда для приведенного при-

мера заголовок и вызов процедуры можно записать следующим образом:
procedure sum(a:at; i:integer; n:integer);
sum(a, n, i);

Для передачи массивов произвольной размерности можно использовать открытые параметры-массивы. Они могут быть параметрами-значениями, параметрами-переменными и параметрами-константами. Заголовок процедуры с открытыми параметрами-массивами имеет вид:

Procedure OpenArray(x: array of OpenArrayType);

Формальный параметр при этом является массивом элементов некоторого типа OpenArrayType с нулевой базой, т.е. Array [0..N-1] of OpenArrayType; где N - количество элементов массива, которое можно определить с помощью стандартной функции High.

Пример: Увеличение вдвое всех элементов массива.

```
Program P603;
const
        n=10; m=20;
type
        t1 = array[1..n] of integer;
        t2 = array[-m..m] of integer;
var
        a : t1;
        b : t2;
        i : integer;
Procedure mult2(var x : array of integer);
var
        i : integer;
begin
        for i:=0 to High(X)-1 do X[i]:=X[i]*2;
end;
begin
        for i:=1 to n do readln(a[i]);
        for i:=-m to m do read(b[i]);
        mult2 (a); {увеличение в 2 раза элементов массива A}
        mult2 (b); {увеличение в 2 раза элементов массива B}
        for i:=1 to n do write(a[i],' ');
        writeln;
        for i:=-m to m do write(b[i],' ');
end.
```

Язык Паскаль допускает наличие у процедур и функций параметров,

которые сами являются процедурами или функциями. Для того чтобы записать такие параметры в заголовке процедуры или функции, они должны иметь некоторый именованный тип. Описание процедурного типа имеет вид:

type <имя типа> = procedure(<описание параметров>);

Например,

type ptype=procedure(var c:real; z:char);

Таким образом, описание процедурного типа представляет собой заголовок процедуры без имени процедуры. Имена параметров, использованные в описании типа, могут быть произвольными (в нашем случае c, z - совершенно случайные идентификаторы).

Функциональный тип описывается следующим образом:

type <имя типа> = function(<описание параметров>):<тип функции>;

Например,

type ftype = function(a:char; b:real):integer;

Не требуется никаких специальных действий, чтобы объявить какую-либо процедуру или функцию как имеющую соответствующий тип. Так, все функции типа integer, имеющие два параметра, первый из которых имеет тип char, а второй - тип real, автоматически будут иметь тип ftype. Но все процедуры и функции, использованные в программе как аргументы, должны быть откомпилированы с опцией {$F+} !

Ниже приведен пример программы, которая будет выводить таблицу значений некоторой функции:

```
Program P604;
type ftype=function(x:real):real;
{$F+}
function f1(x:real):real;
        begin
        f1:=2*cos(x)+1;
        end;

function f2(x:real):real;
        begin
        f2:=exp(x)+cos(x);
        end;
{$F-}
procedure c_func(xmin, xmax:real; n:integer; f:ftype; name:string);
        var
                h, x:real;
                i: integer;
        begin
```

```
        writeln(name);
        h:=(xmax-xmin)/n;
        for i:=0 to n do
                begin
                x:=i*h+xmin;
                writeln(x, f(x));
                end;
        end;
begin
c_func (0,1,10,f1,'функция 1');
c_func (0,10,10,f2,'функция 2');
end.
```

В теле подпрограммы известны (доступны) все объекты, описанные в объемлющем блоке, в том числе и имя самой подпрограммы. Таким образом, внутри тела подпрограммы возможен вызов самой подпрограммы. Процедуры и функции, использующие вызовы "самих себя" называются рекурсивными. Допустима также косвенная рекурсия, при которой, например, процедура A вызывает процедуру B, та, в свою очередь, вызывает C, которая вызывает первоначальную процедуру A.

Рекурсивность часто используется в математике. Так, многие определения математических формул рекурсивны. В качестве примера можно привести формулу вычисления факториала:

$$n! = \begin{cases} 1, \textit{если } \mathrm{n} = 0 \\ n \cdot (n-1)!, \textit{если } \mathrm{n} > 0 \end{cases}.$$

Видно, что для вычисления каждого последующего значения нужно знать предыдущее. В Паскале рекурсия записывается так же, как и в формулах. Для сравнения рассмотрим реализации функции вычисления того же:

```
function fact(n:integer):integer;
begin
        if n=0 then
                        fact:=1
                else
                        fact:=n*fact(n-1);
end;
```

Если в функцию передаются $n>0$, то происходит следующее: запоминаются известные значения членов выражения в ветви else (т.е. n), а для вычисления неизвестных вызывается та же функция fact, но с "предшествующим" аргументом. При этом вновь запоминаются (но в другом месте памяти!) известные значения членов и происходят вызовы. Так происходит

до тех пор, пока выражение не станет полностью определенным (в нашем примере - это присваивание в ветви then), после чего алгоритм начинает "раскручиваться" в обратную сторону, изымая из памяти "отложенные" значения. Поскольку при этом на каждом очередном шаге все члены выражений уже будут известны, через *n* таких обратных шагов мы получим конечный результат.

Необходимым для работоспособности рекурсивных процедур является наличие условия окончания рекурсивных вызовов. Например, проверка значения изменяющегося параметра. Действия, связанные с такой проверкой, уже не могут содержать рекурсивных вызовов. Если это условие не будет выполняться, то глубина рекурсии будет бесконечной, что неизбежно приведет к аварийному останову программы.

Порядок выполнения работы

1. Изучить приемы работы с подпрограммами в языке Паскаль.
2. Разработать алгоритм решения задачи (в соответствии с вариантом) и выбрать вид подпрограммы, которая будет использоваться.
3. Разработать программу на языке Паскаль, с обязательным использованием подпрограммы. Обмен данными с подпрограммой организовать с обязательным использованием параметров.
4. Подготовить отчет.

Варианты заданий

Вариант 6.1

Вычислить

$$y = \sqrt[4]{1 - \frac{\cos^4 t}{4}} + \sqrt[5]{1 + \frac{\operatorname{arctg} t}{2}}.$$

Для вычисления корней с точностью E=0.0001 использовать следующий ряд Тейлора:

$$(1+x)^a = 1 + \frac{a}{1!}x + \frac{a(a-1)}{2!}x^2 + \frac{a(a-1)(a-2)}{3!}x^3 + \ldots \qquad \left(|x| \le 1, a > 0\right).$$

Вариант 6.2

Даны три матрицы. Вывести на экран ту из них, в которой наибольшее число строк, содержащих одни нули (если таких матриц несколько, вывести их все).

Вариант 6.3

Написать программу, содержащую процедуру, которая удаляет указанный элемент из числового массива. При этом все элементы, лежащие ниже удаляемого, поднимаются вверх. Последний элемент массива обнуляется. С помощью указанной процедуры удалить все вхождения наибольшего и наименьшего элементов.

Вариант 6.4

Даны координаты вершин двух прямоугольников, стороны которых параллельны осям координат. Даны координаты десяти точек. В какой из прямоугольников попадет больше точек.

Вариант 6.5

Написать программу, содержащую процедуру, которая вставляет элемент в указанное место числового массива. При этом все элементы, начиная с указанного, передвигаются на одну позицию вниз. Последний элемент массива может быть потерян. С помощью указанной процедуры вставить три элемента, номера и значения которых ввести с клавиатуры.

Вариант 6.6

Составить программу поиска максимальных элементов во всех строках матрицы. Нахождение максимального элемента вектора оформить в виде подпрограммы.

Вариант 6.7

Составить программу, содержащую подпрограмму для перемножения матриц. Предусмотреть контроль правильности исходных данных.

Вариант 6.8

Даны два символьных массива. В массиве, который содержит наибольшее число цифр: заменить все буквы символом "*", а во втором - символом "+".

Вариант 6.9

Даны внешние и внутренние радиусы двух колец, центры которых находятся в начале координат. Даны координаты десяти точек. В какое из колец попадет больше точек.

Вариант 6.10

Даны два массива чисел. Найти сумму элементов того массива, в котором больше положительных чисел.

Вариант 6.11

Определить каких элементов в массиве больше - цифр или латинских строчных букв. Подсчет количества элементов заданного вида оформить в виде подпрограммы.

Вариант 6.12

Составить программу для вычисления площади пятиугольника, заданного координатами вершин. Вычисление расстояния между двумя точками на плоскости оформить в виде подпрограммы.

Вариант 6.13

Написать программу, содержащую процедуру, которая реализует циклический сдвиг влево элементов одномерного массива на заданное количество позиций. В основной программе с помощью указанной процедуры осуществить циклический сдвиг элементов массива на 1, 3 и 4 позиций.

Вариант 6.14

Даны две матрицы (порядок не выше третьего). Транспонировать ту из них, определитель которой больше.

Вариант 6.15

Даны координаты вершин нескольких треугольников. Определить номер треугольника, имеющего наибольшую высоту.

Вариант 6.16

Даны два массива чисел. Расположить их элементы по возрастанию. Сформировать новый массив, включив в него те элементы первого массива, которых нет во втором, и те элементы второго массива, которых нет в первом. Расположить элементы третьего массива по возрастанию.

Вариант 6.17

Дан массив натуральных чисел. Определить, есть ли в нем: а) два подряд идущих одинаковых числа; б) три подряд идущих одинаковых числа; в) **k** подряд идущих одинаковых чисел.

Вариант 6.18

Даны три массива символов. Определить какие символы входят в состав всех трех массивов.

Вариант 6.19

Сформировать массивы **A** и **B**, элементы которых являются квадратами и кубами последовательности натуральных чисел соответственно. Сформировать массив **C**, в который включить все элементы массивов **A** и **B,** расположив их по возрастанию.

Вариант 6.20

Даны два символьных массива. Наиболее часто встречающийся символ в первом массиве (считать, что такой символ один) поменять на наиболее часто встречающийся символ во втором массиве, а наиболее часто встречающийся символ во втором массиве поменять на наиболее часто встречающийся символ в первом массиве.

Вариант 6.21

Дан массив целых чисел. Сформировать наибольшее возможное число из цифр максимального и минимального элементов этого массива.

Вариант 6.22

Даны два массива целых чисел. Определить наибольший общий делитель для всех элементов каждого из массивов. Делители вывести их на экран в порядке возрастания.

Вариант 6.23

Даны два массива целых чисел. Определить наименьшее общее кратное для всех элементов каждого из массивов. Вывести наименьшие общие кратные на экран в порядке убывания.

Вариант 6.24

Даны два массива чисел. Расположить элементы в первом массиве по возрастанию, а во втором по убыванию, используя одну процедуру сортировки.

Вариант 6.25

Написать программу, содержащую процедуру, которая удаляет из матрицы указанную строку или столбец.

Вариант 6.26

Написать программу, содержащую процедуру, которая вставляет в матрицу строку или столбец с заданным номером.

Вариант 6.27

Даны два массива чисел. Найти в них минимальные элементы. Если они равны, то исключить их из массивов. Процедуру повторять до тех пор, пока минимальные элементы массивов не окажутся разными.

Вариант 6.28

Даны два массива целых чисел. Какой из них содержит наибольшее простое число?

Вариант 6.29

Дан символьный массив. Определить, используя подпрограмму, какие символы в массиве встречаются 2 раза, 3 раза, **k** раз?

Вариант 6.30

Дан массив чисел упорядоченных по возрастанию. Написать процедуру, позволяющую добавлять в массив числа без нарушения упорядочения.

Лабораторная работа № 7

Файлы

Цель работы: изучение основных приемов работы с файлами в Паскале.

Любой язык программирования должен содержать средства для организации хранения информации на внешних запоминающих устройствах и доступа к этой информации. В программе на Паскале осуществляется обмен данными с внешней средой посредством файлов. В отличие от данных других типов, файлы размещаются не в оперативной памяти, а на внешних устройствах. В качестве внешних устройств могут использоваться накопители на магнитных дисках, клавиатура, дисплей, печатающее устройство, а также любые другие устройства, которые могут быть подключены к компьютеру. Файлы на магнитных дисках обычно называют дисковыми файлами.

Файл - это структура данных, представляющая собой последовательность элементов одного типа. Размер файла (число элементов) не фиксируется при его описании. Элементы файла обрабатываются, как правило, последовательно один за другим с использованием, так называемого, указателя текущей позиции файла. В каждый конкретный момент времени непосредственно доступен только один элемент, определяемый указателем текущей позиции. Всякий раз, когда выполняется выборка элемента из файла (чтение файла) или помещение элемента в файл (запись в файл), значение указателя текущей позиции автоматически переустанавливается на следующий элемент.

Нумерация элементов файла осуществляется с помощью целых положительных чисел, начиная с нуля: 0,1,2,3,...

Для дисковых файлов обработку элементов можно выполнять также в произвольном порядке. Это возможно благодаря тому, что все элементы имеют одинаковую длину и по номеру элемента легко рассчитать его местоположение на диске. Доступ к нужному элементу осуществляется путем соответствующей установки указателя текущей позиции файла.

В любом случае создание файла может выполняться только последовательно.

Описание типа имеет для файлов следующий формат:

file of <тип элементов>

Тип элементов файла называется базовым типом файла. Он может быть любым типом данных, кроме файлового.

Пример:

```
type tx:array[1..5] of integer;
var
```

```
x1: file of tx;
x2: file of real;
```

Переменные файлового типа описываются в разделе описания переменных, как и все другие переменные. Таким образом, x1 - это файл, состоящий из массивов типа tx; x2 - файл, состоящий из данных вещественного типа.

Файловые переменные не могут использоваться в выражениях или в операторах присваивания. Для работы с файлами существует группа стандартно определенных процедур и функций. Каждая из них имеет параметр "файловая переменная", указывающий файл, над которым должна быть выполнена соответствующая операция.

Перед использованием файловой переменной она должна быть связана с внешним файлом с помощью процедуры Assign. Процедура Assign предназначена для установления связи между конкретным физическим файлом на магнитном носителе и переменной файлового типа, которая будет являться представителем этого файла в программе. Эта процедура имеет два параметра.

assign(<имя файловой переменной>,<строковое выражение>);

Первый параметр - имя файловой переменной, второй параметр - строковое выражение, образующее литеральное имя файла. Имя файла строится по правилам принятым в MS DOS.

После выполнения процедуры assign обращение к файловой переменной в операциях над файлами будет означать выполнение операции над соответствующим файлом на внешнем устройстве.

Пример:

```
var
        f: file of real;
...
        assign (f,'p21.dat');
```

То есть файловая переменная f связывается с файлом P21.dat, расположенном в текущем каталоге.

Когда связь с внешним файлом установлена, для подготовки ее к операции ввода или вывода файловая переменная должна быть "открыта". Существующий файл можно открыть с помощью процедуры reset или rewrite.

Формат:

reset (<имя файловой переменной>);

rewrite (<имя файловой переменной>);

Файловая переменная, указывается в качестве параметра, должна быть уже связана с конкретным дисковым файлом с помощью assign. Под открытием в данном случае поминается поиск файла на диске и установка

текущего указателя файла на его начало (т.е. на нулевой элемент). Разница между этими двумя процедурами заключается в начальных действиях с файлом. Процедура reset предполагает, что открываемый дисковый файл существует, в противном случае возникает ошибка. Процедура rewrite допускает, что открываемый файл может не существовать; в этом случае она создает заданный файл. Если же файл существует, то rewrite очищает его. В любом случае текущий указатель файла устанавливается на его нулевой элемент.

Пример:

```
reset (t);
```

или

```
rewrite (t);
```

Обычно доступ к файлам организуется последовательно, то есть когда элемент считывается с помощью стандартной процедуры read или записывается с помощью стандартной процедуры write, текущая позиция файла перемещается к следующему по порядку элементу файла.

В отличие от многих других процедур read и write могут вызываться с различным числом параметров, и эти параметры могут иметь различные типы.

Процедура read предназначена для чтения значений из файла в программу:

read (<имя файловой переменной>,<список переменных>);

Первым стоит имя файловой переменной, к которой была применена одна из операций открытия (reset или rewrite). Далее должны следовать переменные, в которые будут помещаться читаемые из файла значения. Тип этих переменных должен совпадать с базовым типом файла из первого параметра.

Выполнение процедуры read происходит следующим образом. Начиная с текущей позиции указателя файла, будут последовательно читаться значения, содержащиеся в файле. Каждое прочитанное значение будет присваиваться очередной переменной из тех, которые указаны в вызове процедуры. После каждого чтения, указатель файла будет смещаться на следующую позицию.

Если в процессе выполнения процедуры read текущий указатель файла будет установлен на позицию, не содержащую информации (то есть будет достигнут конец файла), то чтение будет прекращено, процедура read завершится преждевременно, и возникнет ситуация "конец файла".

Процедура write имеет обратное назначение, позволяя записывать в файл информацию из программы:

write (<имя файловой переменной>,<список переменных>);

Первый параметр этой процедуры - файловая переменная откроется процедурой reset или rewrite. Далее следует список выражений, тип которых совпадает с базовым типом файла из первого параметра. Выполнение процедуры write сводится к следующему. Значение очередного выражения будет помещено в файл в место, отмеченное текущим указателем. После этого текущий указатель будет передвинут на одну позицию и действия повторятся для следующего выражения из списка параметров вызова процедуры write.

Когда программа завершает обработку файла, он должен быть закрыт с помощью стандартной процедуры close:

close (<имя файловой переменной>);

После закрытия файла связанный с ним внешний файл обновляется. Файловая переменная может быть связана с другим файлом посредством процедуры assign.

В Паскале есть функция, позволяющая применять последовательный порядок чтения и записи - это процедура seek:

seek (<имя файловой переменной>,<выражение целого типа>);

Процедура seek позволяет явно изменить значение текущего указателя, установив его на элемент файла с заданным номером. Процедура имеет два параметра: переменную файлового типа и целого. Второй параметр понимается как порядковый номер элемента файла, на который необходимо установить указатель.

После выполнения процедуры seek дальнейшие операции чтения или записи будут проводиться, начиная с установленной позиции указателя.

С процедурой Seek связана и процедура

Truncate(<имя файловой переменной>);

позволяющая удалить из файла все записи, начиная с текущей и заканчивая последней, т.е. “отсечь конец” открытого файла.

Функции FileSize и FilePos позволяют получить дополнительную информацию о файле, который указывается в качестве их единственного параметра. Функция FileSize возвращает общее число элементов файла, а функция FilePos - номер элемента, на который установлен текущий указатель.

Пример:

Seek(f,FilePos(f)+1); - пропуск одного элемента

Seek(f,0); - установка указателя на начало файла

Seek(f,FileSize(f)) - установка указателя непосредственно за последним элементом файла; это может служить исходной позицией для добавления элементов к файлу.

Функция EOF вызывается с одним параметром - именем файловой переменной и возвращает логическое значение true или false в зависимости от того, достигнут ли конец файла. Если файл используется для чтения, то возникновение ситуации "конец файла" (и, соответственно, значение true, возвращенное функцией EOF) означает, что все элементы файла прочитаны. При записи в файл истинность функции EOF будет означать, что очередная операция записи поместит информацию в конец данного файла.

Пример: while not EOF(F) do read(f,x);

В качестве примера рассмотрим программу, которая из файла целых чисел выбирает положительные и помещает их в новый файл.

```
Program P701;
var
        f1, f2: file of integer; {описание файловых переменных}
        x: integer;
begin
        Assign (f1,'P01.dat'); {f1 связывается с файлом P01.dat }
        reset(f1);   {открывается существующий файл f1}
        Assign (f2,'P02.dat'); {f2 связывается с файлом P02.dat }
        rewrite(f2);   {открывается новый файл f2}
        while not EOF(f1) do {пока не завершится файл f1}
                begin
                read(f1,x);  {чтение элемента файла f1}
                if x>0 then write(f2,x); {запись положительного элемента в файл
f2}
                end;
        close(f1);   {закрывается файл f1}
        close(f2);   {закрывается файл f2}
end.
```

Для обработки текстовой информации в Паскале имеются специальные средства. Как уже говорилось выше файл - это структура данных, представляющая собой последовательность элементов одинакового типа. Текстовый файл - это файл, состоящий из строк символов. Каждая строка в файле заканчивается маркером конца строки. По той причине, что длина строки в текстовом файле - величина переменная, вычислить положение той или иной строки в файле очень сложно. Поэтому текстовые файлы обрабатываются только последовательно.

Представителем текстового файла в программе на Паскале является файловая переменная, которая должна быть описана с указанием стандартного типа text:

var <имя>: text;

Пример: var ff:text;

Работа с текстовыми файлами организуется построчно, причем характер чтения и записи является строго последовательным. Кроме того, для одного текстового файла нельзя одновременно производить операции и ввода, и вывода.

Для текстовых файлов используются те же операторы assign, rewrite, reset. Но существует ограничение: если файл открыт с помощью процедуры reset, то он может использоваться только для чтения, а если с помощью процедуры rewrite - то только для записи. Открыть существующий текстовый файл для записи можно с помощью процедуры Append. Процедура имеет один параметр - имя файловой переменной.

Append (<имя файловой переменной>);

Эта процедура не очищает файл, а только устанавливает указатель файла в его конец. Процедуру Append удобно использовать, когда необходимо добавить новые строки в конец уже существующего файла.

Чтение из текстового файла и запись в текстовый файл производятся с помощью операторов read и write. Но здесь необходимо понимать, что текстовый файл по определению содержит символьную информацию, поэтому при записи значения других типов будут преобразовываться в символьное представление и в таком виде записываться в очередную строку текстового файла. Аналогично, при чтении из текстового файла очередная часть текущей строки будет пониматься как символьное представление значения, тип которого определяется типом очередной переменной из процедуры read.

Помимо процедур read и write, для текстовых файлов имеются две их модификации - процедуры readln и writeln. Эти процедуры осуществляют те же действия, что и соответствующие процедуры read и write, но после операции чтения и записи производят переход к следующей строке текстового файла. Заметим, что операция read автоматически переходит к следующей строке только в случае исчерпания текущей строки; процедура readln позволяет совершить этот переход, не дожидаясь исчерпания строки.

Следить за исчерпанием текстового файла можно с помощью процедуры EOF.

Пример. Существует текстовый файл f1.txt. Необходимо сформировать файл f2.txt, содержащий только те строки из файла f1.txt, в первой позиции которых находится символ '*'.

```
Program P702;
var
        f1, f2: text; {описание файловых переменных}
        s: string;
begin
        Assign (f1,'P01.txt'); {f1 связывается с файлом P01.txt }
        reset(f1);   {открывается файл f1 для чтения}
        Assign (f2,'P02.txt'); {f2 связывается с файлом P02.txt }
        rewrite(f2);   {открывается файл f2 для записи}
        while not EOF(f1) do {пока не завершится файл f1}
                begin
                read(f1,s);  {чтение элемента файла f1}
                if s[1]='*' then write(f2,s); {запись искомого элемента в файл f2}
                end;
        close(f1);   {закрывается файл f1}
        close(f2);   {закрывается файл f2}
end.
```

Для работы с файлами в Паскале имеются и другие возможности.

Процедура
ReName(<имя файловой переменной>, <новое имя файла>);
позволяет переименовать физический файл, связанный в программе с файловой переменной, присвоив ему <новое имя файла>, которое должно быть константой, переменной или выражением строкового типа.

Пример:

```
Assign(f,'p.dat'); {Связать переменную f с файлом p.dat}
ReName(f,'a.dat'); {Переименовать p.dat в a.dat}
```

Процедура
Erase(<имя файловой переменной>);
удаляет (стирает) с внешнего носителя физический файл, связанный с файловой переменной.

Пример:

```
Assign(f,'p.dat'); {Связать переменную f с файлом p.dat}
Erase(f); {Удалить p.dat}
```

Процедуры переименования и удаления применяются только к закрытым файлам.

Попытка переименования или удаления несуществующего файла вызовет прерывание выполнения программы.

Порядок выполнения работы

1. Изучить приемы работы с файлами в языке Паскаль.
2. Разработать алгоритм и программу формирования файла (в соответствии с вариантом) и сформировать файл на диске.
3. Разработать алгоритм и программу обработки файла (в соответствии с вариантом).
4. Разработать (при необходимости) алгоритм и программу проверки правильности обработки файла.
5. Подготовить отчет.

Варианты заданий

Вариант 7.1

Составить программы, с помощью которых:

- сформировать файл из целых чисел, которые должны быть введены с клавиатуры;

- заменить все отрицательные числа в сформированном файле нулями.

Вариант 7.2

Составить программы, с помощью которых:

- сформировать файл из символов, которые должны быть введены с клавиатуры;

- заменить символы +, -, *, / в сформированном файле символами **55**.

Вариант 7.3

Составить программы, с помощью которых:

- сформировать файл из целых чисел, которые должны быть введены с клавиатуры;

- исключить из файла все компоненты, находящиеся между наибольшей и наименьшей компонентами.

Вариант 7.4

Составить программы, с помощью которых:

- сформировать файл из целых чисел, которые должны быть введены с клавиатуры;

- упорядочить элементы файла так, чтобы сначала следовали все положительные компоненты, а затем все остальные.

Вариант 7.5

Составить программы, с помощью которых:

- сформировать файл из символов, которые должны быть введены с клавиатуры;

- после каждого символа + добавить символ *.

Вариант 7.6

Составить программы, с помощью которых:

- сформировать файл из вещественных чисел, которые должны быть введены с клавиатуры;

- исключить из файла компоненты, которые меньше предшествующей компоненты.

Вариант 7.7

Составить программы, с помощью которых:

- сформировать файл, элементами которого являются массивы вещественных чисел, введенных с клавиатуры;

- исключить из файла массивы, сумма элементов которых является наименьшей.

Вариант 7.8

Составить программы, с помощью которых:

- сформировать файл, элементами которого являются массивы вещественных чисел, введенных с клавиатуры;

- добавить в файл компоненту, номер и значения которой ввести с клавиатуры.

Вариант 7.9

Составить программы, с помощью которых:

- сформировать файл, элементами которого являются массивы вещественных чисел, введенных с клавиатуры;

- упорядочить компоненты файла так, чтобы сначала следовали все массивы с положительными максимальными элементами, а затем все остальные.

Вариант 7.10

Составить программы, с помощью которых:

- сформировать файл, элементами которого являются массивы вещественных чисел, введенных с клавиатуры;

- во всех компонентах файла исключить максимальные элементы.

Вариант 7.11

Составить программы, с помощью которых:

- сформировать файл из целых чисел, которые должны быть введены с клавиатуры;

- исключить из файла все компоненты, совпадающие с предыдущей компонентой.

Вариант 7.12

Составить программы, с помощью которых:

- сформировать файл из записей, содержащих различные даты;

- исключить из файла все компоненты, содержащие весенние даты.

Вариант 7.13

Составить программы, с помощью которых:

- сформировать файл из записей, содержащих различные даты;

- найти самую позднюю дату.

Вариант 7.14

Составить программы, с помощью которых:

- сформировать файл из записей, содержащих сведения о книге (фамилия автора, название, год издания);

- исключить все книги с названием «Информатика».

Вариант 7.15

Составить программы, с помощью которых:

- сформировать файл из записей, содержащих сведения о книге (фамилия автора, название, год издания);

- упорядочить компоненты файла так, чтобы сначала следовали все книги Петрова, а затем все остальные.

Вариант 7.16

Составить программы, с помощью которых:

- сформировать файл из записей, содержащих сведения о книге (фамилия автора, название, год издания);

- определить фамилии авторов, книги которых были изданы после 1980 года.

Вариант 7.17

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- вывести на экран строки наименьшей длины.

Вариант 7.18

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- вывести на экран строки содержащие слово «стол».

Вариант 7.19

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- вывести на экран строки в порядке возрастания их длины.

Вариант 7.20

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- в файл после каждой строки, содержащей слово «бит», добавить строку, состоящую из десяти символов ”*”.

Вариант 7.21

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- вывести на экран строки с наибольшим количеством слов.

Вариант 7.22

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- в файле из каждой строки удалить второе слово.

Вариант 7.23

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- сформировать новый файл, в который включить строки, состоящие из трех слов.

Вариант 7.24

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- в файле из каждой строки удалить второе слово.

Вариант 7.25

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- вывести на экран строки, не содержащие слово «Паскаль».

Вариант 7.26

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- вывести на экран строки в алфавитном порядке.

Вариант 7.27

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- перед каждым словом «дом» добавить слово «большой».

Вариант 7.28

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- заменить каждое слово «синий» словом «зеленый».

Вариант 7.29

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- подсчитать количество слов, содержащих хотя бы одну букву «м».

Вариант 7.30

Составить программы, с помощью которых:

- сформировать текстовый файл из строк, которые должны быть введены с клавиатуры;

- подсчитать количество слов, содержащих две буквы «м».

Рекомендуемая литература

1. Епанешников, А.М. Программирование в среде Turbo-Pascal 7.0 / А.М. Епанешников, В.А. Епанешников. - М.:, Диалог МИФИ, 1995. – 288 с.

2. Культин, Н.Б. Turbo-Pascal в задачах и примерах. – СПб.: БХВ-Петербург, 2006. – 256 с.

3. Немнюгин, С.А. Turbo-Pascal. – СПб.: Издательство "Питер", 2000. – 496 с.

4. Перминов, О.Н. Программирование на языке Паскаль. - М.: Радио и связь, 1988. - 224 с.

5. Пильщиков,_В.Н. Сборник упражнений по языку Паскаль. - М.: Наука, 1989. – 160 с.

6. Поляков, Д.Б. Программирование в среде Турбо-Паскаля (версия 5.5) /. Д.Б. Поляков, Н.Ю. Круглов. - М.: изд. МАИ, 1992. - 586 с.

7. Фаронов, В.В. Turbo-Pascal. – СПб.: БХВ-Петербург, 2004. – 1056 с.

СОДЕРЖАНИЕ

Лабораторная работа №1. Работа в среде Turbo Pascal 3
Лабораторная работа №2. Программа линейной структуры 8
Лабораторная работа №3. Программирование процессов ветвящейся структуры 23
Лабораторная работа №4. Программирование процессов циклической структуры 38
Лабораторная работа №5. Массивы 53
Лабораторная работа №6. Функции и процедуры 60
Лабораторная работа №7. Файлы 75
Рекомендуемая литература 89