АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ПАСКАЛЬ

Методические указания к лабораторным работам

Лабораторная работа № 1

ПРОГРАММИРОВАНИЕ ФОРМУЛ; ОПЕРАТОРЫ ВВОДА И ВЫВОДА

ЦЕЛЬ РАБОТЫ: Познакомиться с базовой алгоритмической структурой «следование»; познакомиться со структурой простых программ, операторами ввода и вывода данных; научиться программировать арифметические и алгебраические вычисления.

Алгоритмизация. В настоящей работе рассматриваются программы, реализующие базовую алгоритмическую структуру «следование», рисунок 1, которая предполагает последовательное (одно за другим) выполнение действий.

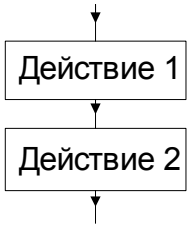


Рисунок 1

Программирование. При реализации указанной структуры на языке Паскаль используются операторы ввода, вывода, присваивания и рассматриваются арифметические действия. В таблицах 1 и 2 приведены обозначения арифметических действий, а также часто употребляемых функций и соответствующие им конструкции в языке Паскаль.

Таблица 1

Арифметические действия в языке PASCAL

Обозначение	Действие	Пример
+	Сложение	a+b
–	Вычитание	a–b
*	Умножение	a*b
/	Деление	a/b
div	Целочисленное деление	a div b
mod	Остаток от целочисленного деления	a mod b

Таблица 2

Математическая функция	Обозначение в языке Pascal	Математическая функция	Обозначение в языке Pascal
ln x	ln(x)	$\sqrt[b]{x} = x^{\frac{1}{b}}$	exp(ln(x)/b)
lg x	ln(x)/ln(10)	sin x	sin(x)
e^x	exp(x)	cos x	cos(x)
a^b	exp(b*ln(a))	tg x	sin(x)/cos(x)
x^2	sqr(x)	arctg x	atan(x)
$\sqrt{x}$	sqrt(x)	|x|	abs(x)

В лабораторной работе используются операторы:

- **присваивания**, имеющие формат: **<Идентификатор>:=<Выражение>;** (вычислить значение *выражения*, стоящего справа от операции **:=** (*присваивания*) и поместить результат в переменную, имя (*идентификатор*) которой стоит слева от знака операции присваивания);
- процедура **Read()** для ввода данных с клавиатуры или из файла;
- процедуры **Write()** и **WriteLn()** для вывода данных на дисплей или в файл.

Ввод данных из файла и вывод в файл будут рассмотрены в лабораторной работе № 6.

Задачи, разбираемые в данной лабораторной работе, имеют следующий алгоритм выполнения:

1) ввод исходных данных с клавиатуры;
2) вычисление формулы;
3) вывод результатов вычислений на дисплей.

При составлении алгоритмов необходимо руководствоваться ГОСТ 19.701-90, выдержка из которого приведена в приложении 1.

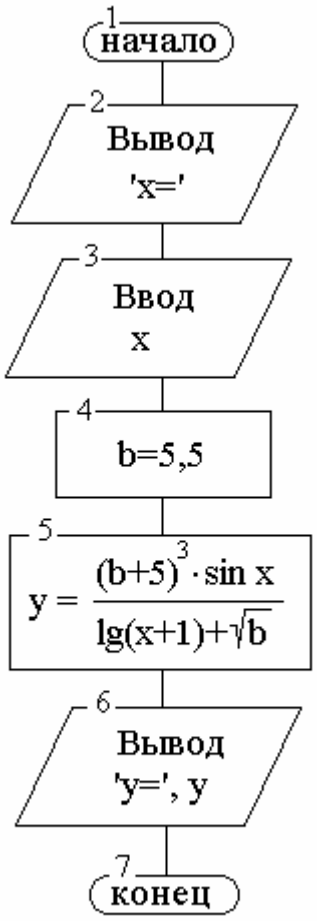


Рисунок 2

Пример 1.1. Написать алгоритм и программу для вычисления формулы

$$y = \frac{(b+5)^3 \cdot \sin x}{\lg(x+1)+\sqrt{b}}$$ для b= 5,5; x=0,1; 0,5.

Схема алгоритма, реализующего поставленную задачу, изображена на рисунке 2. Здесь отметим, что прежде, чем ввести данные с клавиатуры (элемент № 3 алгоритма), необходимо побудить к этому оператора (элемент № 2). Ниже приведен текст программы.

```
program lab1;
const B=5.5;
var y, x:Real;
begin
   Write('x=');
   Read(x);
   y:=exp(3*ln(B+5))*sin(x)/(ln(x+1)/ln(10)+sqrt(B));
   WriteLn('y=',y:6:2);
end.
```

Программирование обычно осуществляется в среде Турбо Паскаль, в которую при работе в дисплейном классе можно попасть через меню пользователя, зайдя предварительно в рабочий каталог. Если соответствующая команда в меню пользователя отсутствует, то необходимо включить в меню программу turbo.exe. При работе в среде Windows можно создать на рабочем столе ярлык для этой программы.

После запуска turbo.exe выбираем пункт меню File → New. В открывшемся окне набираем текст программы. Сохраняя программу клавишей F2, в появившемся окне

вводим имя файла с составленной нами программой. Запуская программу из примера 1.1 на выполнение (комбинацией клавиш Ctrl+F9) два раза, вводим с клавиатуры соответственно **0.1** и **0.5**. Просмотр результатов выполнения программы осуществляем комбинацией клавиш Alt+F5. Возврат в окно с текстом программы после просмотра результатов – нажатие любой клавиши.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Для данных из таблицы 3 написать алгоритм и программу для вычисления формулы для *x* или *t*, введенных с клавиатуры. Произвести тестирование программы с помощью имеющегося математического пакета или калькулятора.

Таблица 3

Вариант	Константы	Переменные	Формула
1	a=3; b= –0,5	x=2; 1,3	$y := a - \frac{\lg(a + x)\cdot x^{b}}{\cos(x\cdot\pi)}$
2	b= 1,5	x=2; 0,3	$y = \frac{b^{5} - \operatorname{tg} x}{\ln x}$
3	a=1,3; b= 2,5	x=2,2; 1,3	$y = \frac{a\cdot b - e^{a\cdot x} + \cos x}{\lg(x) + 1}$
4	a=1,5; b= –100	x=0,8; 1,3	$y = \frac{\left(\frac{a}{b} - 3^{a\cdot x}\right)\cdot\cos x}{\sqrt{x + 2}}$
5	a=1; b= 2	x= –0,5; 0,5	$y = \frac{\left(a + \sqrt[3]{x + 3}\right)\cdot(x + b)}{\ln b}$

Продолжение таблицы 3

Вариант	Константы	Переменные	Формула
6	a=π; b= 5	x=6,0; 6,5	$y = \frac{b \cdot e^{(x-b)}}{tg(5 \cdot x)} + \frac{a}{b}$
7	a=π/2; b= –0,9	x=1; 2	$y = \frac{x^b - a^x}{b^2 + a^x} \cdot 3$
8	a=1,1; b= 4	x=–3; 3	$y = \frac{\ln(\lvert x + a \rvert) b}{\sqrt{a^b} - \sin b}$
9	a=2	x=2; 2,5	$y = \frac{\sin^2 x - a^x}{\sqrt{x \cdot \pi - 4} - a}$
10	a=3	x=1,5; 3,5	$y = \frac{\sqrt[3]{x} + \frac{1}{a}}{\lg\left(\left\lvert \sqrt[3]{x} + \frac{1}{a} \right\rvert\right)}$
11	a=4	x= –6; –5	$y = \frac{\frac{1}{\sqrt{\lvert x + 1 \rvert + a}} + \frac{1}{a}}{arctg(a - x)}$
12	a=5	x= 10; 0	$y = \frac{\lg^2(x + a) - \lvert x + a \rvert}{\cos^2(x + a)}$

Продолжение таблицы 3

Вариант	Константы	Переменные	Формула
13	a=0,5; b= –3	t=0,9	$z := \dfrac{\dfrac{1}{\cos(t)} + \dfrac{1}{\cos^3\left(t^2\right)} + t}{e^{2\cdot t\cdot b}}$
14	a= –π; b= π	t=1,7	$z := \dfrac{\sin(a\cdot t) + \left\lvert t^{10}\right\rvert}{\cos(b\cdot t) + \dfrac{1}{5}}$
15	a= 14; b= 200	t=10	$z = \dfrac{\ln\left(t^{2,5} + a\right) - \sqrt{\lg\left(t^2\right)}}{\operatorname{arctg}\dfrac{t}{b}}$
16	a= 50; b= π/3	t=5,5	$z = \dfrac{e^{0,01\cdot a} - \cos(b\cdot t)\cdot\lvert t\rvert}{\lg^5 t + \dfrac{1}{t}}$
17	a= π; b= 2π	t=0,8	$z = \dfrac{e^{\sin(a\cdot t)} + \cos(b\cdot t)}{5\cdot\cos^2(a\cdot t) - a} + \dfrac{1}{t}$
18	a= –2; b= 3	t=4	$z = \dfrac{\lg t + \lvert t\rvert}{a\cdot t^2}\cdot b + t^{2,5}$
19	a= –1; b= 2,3	t=8	$z = \left\lvert\dfrac{\cos(2\cdot t)}{a + \sin t}\right\rvert + t^2\cdot\sqrt{\sin t}$

Продолжение таблицы 3

Вариант	Константы	Переменные	Формула
20	a= –π; b= π	t=5	$z = \frac{t^3}{\cos(a \cdot t) + \sin(b \cdot t)} + \left\lvert t \cdot e^t \right\rvert$
21	c= –1; d= 1,5	x= 0; 1	$y = \sin\left(x\right) \cdot c + \frac{\lg\left(d + x\right)}{\sqrt{d}}$
22	c= 2; d= –2,5	x= 2; 3	$y = \frac{\frac{\operatorname{tg} x}{\ln x} + \frac{x}{c}}{\left\lvert d \cdot x \right\rvert}$
23	c= 10; d= –9	x= 2,2; 0	$y = \operatorname{tg}\left(x\right) \cdot \frac{\lg\left(\left\lvert d + x \right\rvert\right)}{\sqrt{c \cdot x} + d}$
24	c= 5; d= 6	x= 3; –4	$y = \frac{\sqrt{\left(x \cdot c\right)^2 + d}}{c^{0.1d} - \lg\left(\left\lvert d \cdot x \right\rvert\right)}$
25	c= –1,5; d= 3	x= 0,5; 1	$y = \frac{x^d \cdot \cos x}{\sqrt{d - x} + c}$
26	c= –3; d= 5π	x= 1,8; 2	$y = \frac{\operatorname{arctg}\left(x \cdot d\right) + \left\lvert x \right\rvert}{\sqrt{d - x} + c}$
27	c= –8; d= 7	x=0,1; 0,3	$y = \frac{\frac{1}{c + x} + e^{\left\lvert \sin x \right\rvert}}{\lg\left(d - x\right)}$

Продолжение таблицы 3

Вариант	Константы	Переменные	Формула
28	c= 1; d= 2	x= 3; 4	$y = \dfrac{\operatorname{tg}\left(c + x\right) - \dfrac{1}{\lg\left(c \cdot x\right)}}{\sqrt[5]{d \cdot x}}$
29	c= 1,7; d= 0,5	x= 1,5; 2	$y = \dfrac{\sin\left(\ln\left(c + d \cdot x\right)\right) + \left(d + x\right)^3}{c \cdot x^2 + d \cdot x - 1}$
30	c= 6; d= 10	x= 0; 0,5	$y = \sqrt[3]{\dfrac{\lg\left(c + x\right) + e^{5 \cdot c}}{d^{c+x} + d}}$

СОДЕРЖАНИЕ ОТЧЕТА

1. Лабораторное задание.
2. Структурная схема алгоритма.
3. Текст программы.

Лабораторная работа № 2

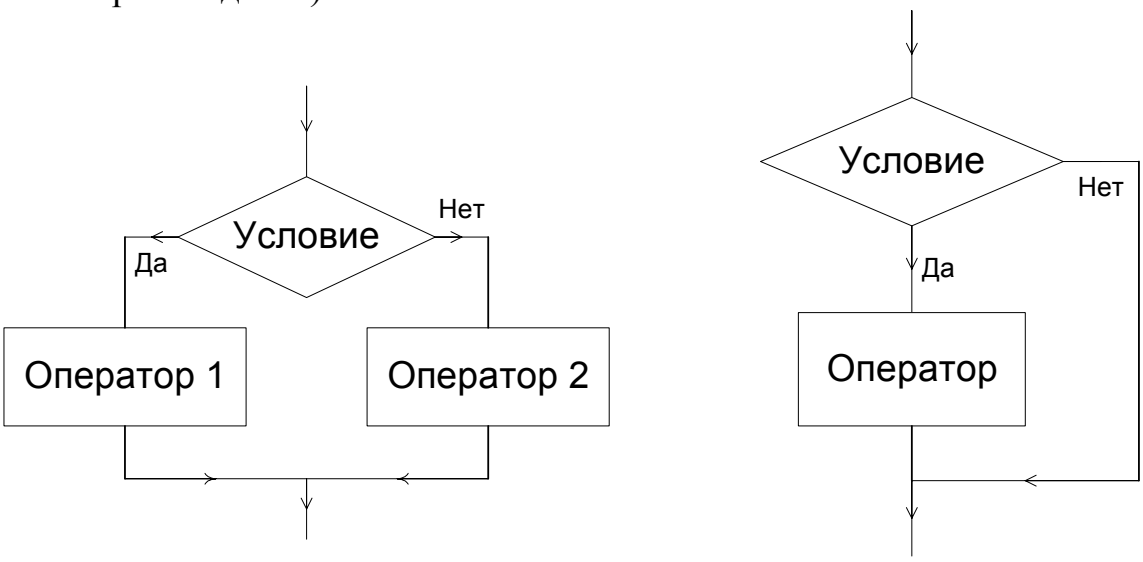
УСЛОВНЫЙ ОПЕРАТОР

ЦЕЛЬ РАБОТЫ: Познакомиться с условным оператором IF.

Алгоритмизация. В настоящей работе рассматривается оператор, реализующий алгоритмические структуры «ветвление», изображенные на рисунке 3.

Ромбические элементы алгоритма означают проверку условия, помещенного внутрь ромба. Если условие выполняется, то осуществляется переход по ветке «да», если не выполняется – по ветке «нет». Таким образом, осуществляется выбор одного из двух вариантов действий. На рисунке а) изображена базовая структура, в которой выполняется либо действие 1 (оператор 1), либо действие 2 (оператор 2). Частным

случаем базовой структуры является структура на рисунке б), в которой действие (оператор) либо выполняется, либо не выполняется (т.е., по ветке «нет» ни каких действий не производится).



а) б)

Рисунок 3

Программирование. В языке Паскаль структуре на рисунке 3а соответствует оператор IF в формате «с иначе»:

IF условие THEN оператор_1 ELSE оператор_2,

ЕСЛИ ТОГДА ИНАЧЕ

а структуре на рисунке 3б – в формате «без иначе»:

IF условие THEN оператор,

где *условие* – выражение логического типа, результат которого есть TRUE (ИСТИНА), если условие выполняется, или FALSE (ЛОЖЬ), если условие не выполняется;

оператор_1, *оператор_2* и *оператор* – любые операторы Турбо Паскаля, могут быть простыми либо составными. Тело составного оператора состоит из нескольких операторов и ограничивается операторными скобками begin, end:

```
begin
  оператор1;
  оператор2;
  оператор3
end
```

составной оператор

Условие в операторе IF может быть как простым, так и сложным, объединяющим простые условия логическими связками OR (или), AND (и), NOT (не).

Пример 2.1. Написать алгоритм и программу вычисления функции y(x) по формуле

$$y=\begin{cases}\dfrac{tg\ x+\lg|x-1|}{2x+5}, & -1\le x\le 0\\ \sqrt[3]{x}, & 0<x\le 1\\ 5\sin 2x, & 1<x\le 2\end{cases}$$

Структурная схема алгоритма (ССА) изображена на рисунке 4.

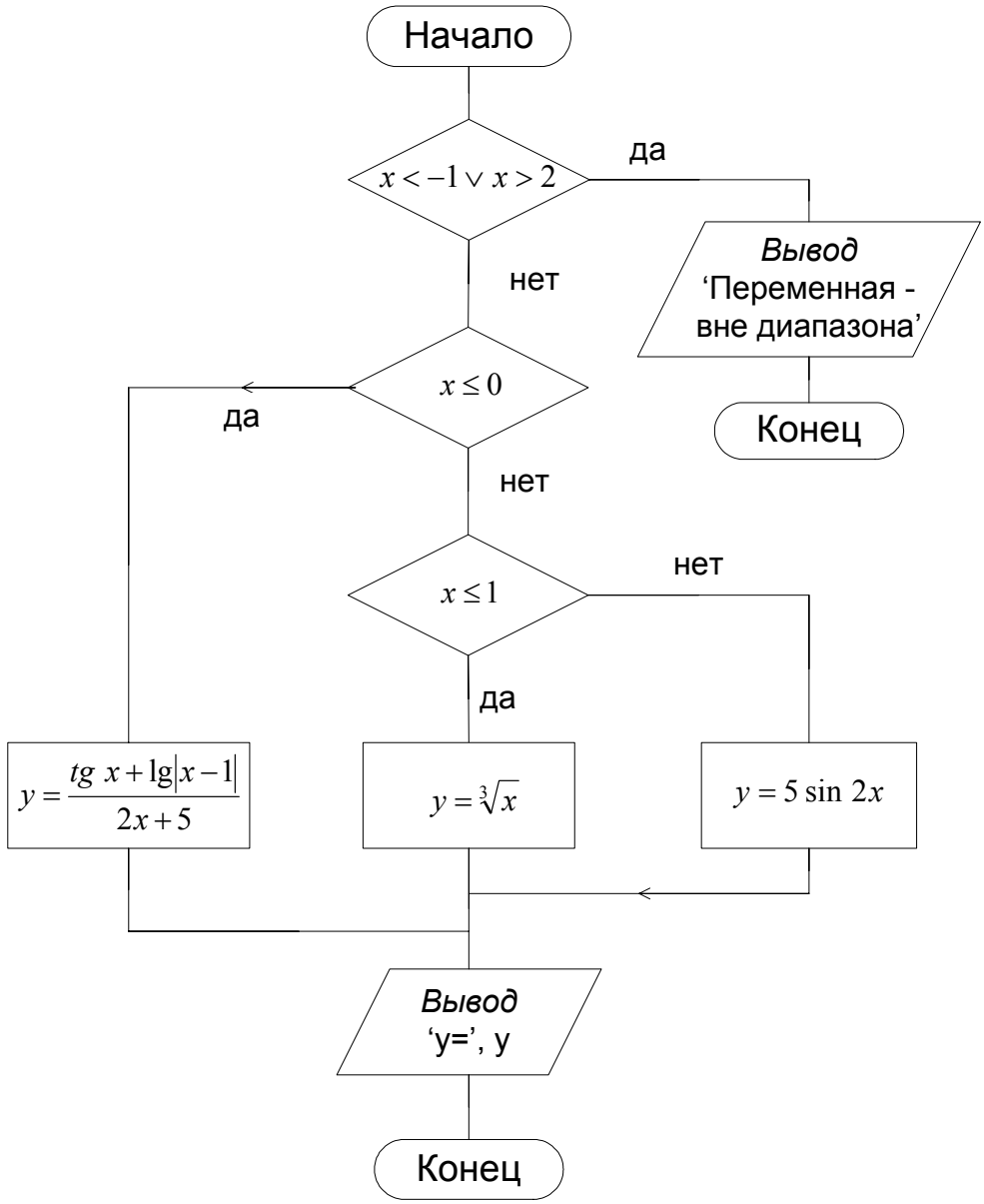


Рисунок 4

```
Program lab2_1;
Var y,x: Real;
Begin
   Write('x=');
   Read(x);
   If ((x<-1) or (x>2))
        Then WriteLn('Variable X is out range')
```

```
        Else
           Begin
              If x<=0 Then y:=(sin(x)/cos(x)+ln(abs(x-
1))/ln(10))/(2*x+5)
                      Else If x<=1 Then y:=exp(ln(x)/3)
                                   Else y:=5*sin(2*x);
              WriteLn('y=',y:6:3)
           End

End.
```

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Написать алгоритм и программу для вычисления значения y из таблицы 4 для x, введенного с клавиатуры. Произвести тестирование программы.

Таблица 4

Номер задания	Формула	Номер задания	Формула
1	$y=\begin{cases}\sin 5x, & -1\le x<0\\ tg\,(1+x), & 0\le x<1\\ \dfrac{1}{1+x}, & 1\le x\le 2\end{cases}$	2	$y=\begin{cases}x, & 1\le x<2\\ 2+\cos x, & 2\le x<3\\ \lg 6x, & 3\le x\le 4\end{cases}$
3	$y=\begin{cases}\dfrac{tg\,x}{2}, & 0\le x<0{,}8\\ \dfrac{\sin x}{x}, & 0{,}8\le x<1{,}3\\ x^3, & 1{,}3\le x\le 2\end{cases}$	4	$y=\begin{cases}4\cdot e^x, & -2\le x<-1{,}4\\ x^2+2x, & -1{,}4\le x<-0{,}4\\ \pi, & -0{,}4\le x\le 0\end{cases}$
5	$y=\begin{cases}arctg\,x, & 0{,}1\le x<0{,}8\\ \cos(x)\cdot\sin(x), & 0{,}8\le x<1{,}3\\ \dfrac{e^x+e^{-x}}{2}, & 1{,}3\le x\le 1{,}8\end{cases}$	6	$y=\begin{cases}\lvert x\rvert, & -0{,}5\le x<0\\ 2+x, & 0\le x<0{,}5\\ 2^x+1, & 0{,}5\le x\le 1\end{cases}$
7	$y=\begin{cases}\dfrac{10}{x+1{,}5}, & 1\le x<3\\ 2^x-2, & 3\le x<5\\ \sqrt{\pi\cdot x}, & 5\le x\le 7\end{cases}$	8	$y=\begin{cases}x\cdot(3-x), & -1\le x<0\\ 1, & 0\le x<1\\ tg\,2x, & 1\le x\le 2\end{cases}$
9	$y=\begin{cases}\dfrac{e^x}{e^{-x}+\pi-1}, & -1\le x<0\\ \dfrac{20\sin x}{(2+x)^3}, & 0\le x\le 1\end{cases}$	10	$y=\begin{cases}\pi\cdot\cos x, & 0\le x<0{,}5\\ x\cdot tg\,x-\ln x, & 0{,}5\le x\le 1\end{cases}$

Продолжение таблицы 4

Номер задания	Формула	Номер задания	Формула
11	$y=\begin{cases}\sqrt[3]{2+x}, & 1\le x<2\\ \lg\left(\dfrac{x-1}{x+1}\right), & 2\le x\le 3\end{cases}$	12	$y=\begin{cases}\lg\left\|\dfrac{1}{x}\right\|, & -2\le x<-1{,}5\\ \dfrac{e^{x}-e^{-x}}{2}, & -1{,}5\le x\le -1\end{cases}$
13	$y=\begin{cases}\sqrt{\left\|0{,}5x^2-4x+2\right\|}, & -0{,}1\le x<0{,}1\\ \dfrac{x^3+0{,}5x^2-4x+2}{\sin x}, & 0{,}1\le x\le 0{,}3\end{cases}$	14	$y=\begin{cases}\dfrac{100\lg x}{(x+1)^4}, & 1\le x<10\\ \sqrt{x+5}-\pi, & 10\le x\le 100\end{cases}$
15	$y=\begin{cases}\dfrac{tg\ x}{x-3}, & -2\le x<0\\ \lg x\cdot\sin x, & 0\le x\le 2\end{cases}$	16	$y=\begin{cases}arctg\ x+\dfrac{\pi}{3}, & 1\le x<2\\ (1-5x)\sqrt{\sin x}, & 2\le x\le 3\end{cases}$
17	$y=\begin{cases}\dfrac{\sin 0{,}1x}{x-2}+\sqrt{x}, & 10\le x<20\\ \cos(x)-x^{3{,}5}, & 20\le x\le 30\end{cases}$	18	$y=\begin{cases}\lg\left\|1+2x\right\|, & -5\le x<0\\ e^{arctg\ x}\cdot\dfrac{\sin x}{1+\dfrac{1}{x}}, & 0\le x\le 5\end{cases}$
19	$y=\begin{cases}\dfrac{tg\ x}{1+tg\ 2x}, & -2\le x\le 1\\ (\lg x)^{2{,}5}, & 1<x\le 4\end{cases}$	20	$y=\begin{cases}\dfrac{\sin x}{x+\cos x}, & 0{,}1\le x\le 1\\ x^3+2x^2+3x+4, & 1<x\le 1{,}9\end{cases}$
21	$y=\begin{cases}1-\dfrac{x}{\sqrt[5]{1+x}}, & 1{,}5\le x\le 2{,}5\\ \dfrac{100(\ln x+\sin x)}{(2x-3)^2}, & 2{,}5<x\le 3{,}5\end{cases}$	22	$y=\begin{cases}5-\dfrac{\lg(3+x)\,x^2}{\cos\pi x}, & 0{,}2\le x\le 0{,}4\\ \dfrac{x^5-tg\ x}{\ln x}, & 0{,}4<x\le 0{,}6\end{cases}$
23	$y=\begin{cases}100\dfrac{5x-e^{3x}+\cos x}{1+\lg x}, & 0{,}2\le x\le 0{,}4\\ (1+x)^6, & 0{,}4<x\le 0{,}6\end{cases}$	24	$y=\begin{cases}\sqrt[6]{1+x}, & 2\le x\le 3\\ \dfrac{\left(\dfrac{2}{3}-3^{2x}\right)\cdot\cos x}{\sqrt{x+2}}, & 3<x\le 4\end{cases}$
25	$y=\begin{cases}arctg(\pi+x), & 2{,}5\le x\le 3{,}3\\ \dfrac{\left(1+\sqrt[3]{x+3}\right)\cdot(x+2)}{\ln 2}, & 3{,}3<x\le 4{,}4\end{cases}$	26	$y=\begin{cases}\dfrac{1}{10+2x+3x^2}, & -1\le x\le 0\\ \dfrac{5e^{x-5}}{1+tg\ 5x}+\dfrac{2}{5}, & 0<x\le 1\end{cases}$
27	$y=\begin{cases}\dfrac{x^3-2^x}{(1+x)^2+2^x}\cdot 3, & 2{,}5\le x\le 5\\ \lg(1+x)+\dfrac{2}{5}, & 5<x\le 7{,}5\end{cases}$	28	$y=\begin{cases}e^{\frac{1}{1+x}}, & 8\le x\le 9\\ \dfrac{3\cdot\ln\left\|x+2\right\|}{\sqrt{2+x}-\sin 3x}, & 9<x\le 10\end{cases}$

29	$y=\begin{cases}\dfrac{1}{\sqrt{\lvert x+1\rvert+2}}+\dfrac{1}{2}, & -2\le x\le -1\\ arctg(2-x), & \\ x, & -1<x\le 0\end{cases}$	30	$y=\begin{cases}x^3+2x^2+x, & -1\le x\le 0\\ \dfrac{\lg^2(1+x)-\lvert x+2\rvert}{\cos^2(x+2)}, & 0<x\le 1\end{cases}$

СОДЕРЖАНИЕ ОТЧЕТА

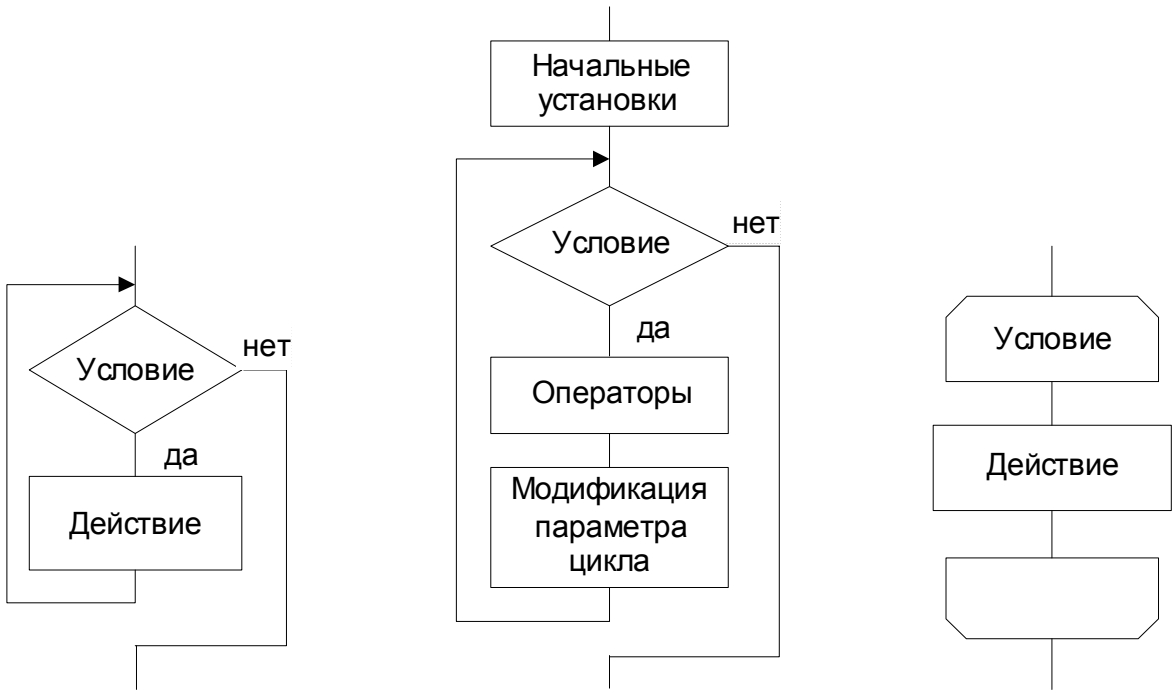
1. Лабораторное задание.
2. Структурная схема алгоритма.
3. Текст программы.

Лабораторная работа № 3

ЦИКЛ С ПРЕДУСЛОВИЕМ

ЦЕЛЬ РАБОТЫ: Изучить цикл с предусловием.

Алгоритмизация. В настоящей работе рассматривается оператор WHILE, реализующий базовую алгоритмическую структуру «цикл-пока», рисунок 5а.



а) б) в)

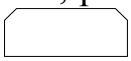
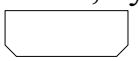
Рисунок 5

Цикл предназначен для многократного выполнения некоторых действий. Число проходов такого цикла, называемых итерациями, может быть заранее не известно. Условие, проверяемое в начале цикла, есть условие входа в цикл или продолжения выполнения цикла. Когда во время работы цикла условие перестает выполняться, происходит выход из цикла и управление передается оператору, следующему за

оператором цикла. Если условие не выполняется при входе в цикл, то цикл не совершит ни одного прохода и управление передается оператору, следующему за оператором цикла.

Заметим, что проверка условия, изображенная на рисунке 5а, осуществляется не с помощью оператора IF, а заложена в самом операторе цикла WHILE.

Схема, представленная на рисунке 5а, является упрощенной. Во-первых, в условии фигурируют одна или несколько переменных, называемых *параметрами цикла*. И, в общем случае, к началу выполнения цикла, специально для его организации, в эти переменные требуется занести определенные значения. То есть, произвести *начальные установки (инициализацию)*, рисунок 5б. Во-вторых, указанное условие после некоторого числа проходов должно перестать выполняться. А значит, параметры цикла в ходе его выполнения должны изменяться. Иначе условие будет выполняться всегда, и цикл будет выполняться «бесконечно». Для того, чтобы последнее не произошло, необходимо внутри цикла изменять его параметры, то есть, производить *модификацию параметров цикла*, рисунок 5б.

Для обозначения цикла с предусловием, рисунок 5а, существует структура, изображенная на рисунке 5в, где элементы  и  обозначают соответственно начало и конец цикла.

Программирование. В языке Паскаль оператор цикла с предусловием записывается следующим образом:

WHILE *условие* DO *оператор*,

где *оператор*, стоящий после DO, составляет тело цикла и может быть простым или составным.

И последнее замечание. В ходе работы цикла проверка условия осуществляется по завершению выполнения оператора, образующего тело цикла.

Пример 3.1. Написать алгоритм и программу вычисления y(x) из Примера 1.1 для $x \in [0{,}1, 1]$ с шагом 0,08. Первое значение y(x) вычислить в левой крайней точке диапазона x=0,1.

ССА изображена на рисунке 6.

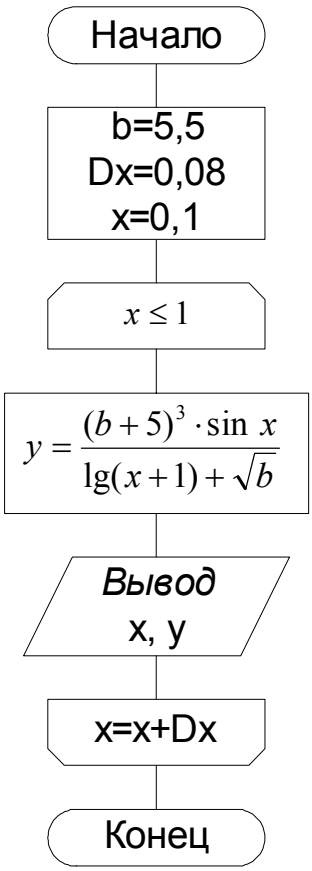


Рисунок 6

```
Program lab3_1;
Const B=5.5;
      Dx=0.08;
Var x,y: Real;
Begin
   x:=0.1;
   While (x<=1) do
     Begin
       y:=(Exp(3*Ln(B+5))*sin(x))/(ln(x+1)/ln(10)+sqrt(B));
       WriteLn('x=',x:4:2,' y=',y:5:1);
       x:=x+Dx;
     End;
End.
```

К о м м е н т а р и и к программе `lab3_1`.

Параметром цикла является переменная `x`. Начальные установки цикла задаются с помощью оператора `x:=0.1`. Модификация параметра цикла осуществляется оператором `x:=x+Dx`.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Написать алгоритм и программу вычисления y(x) или z(t) из лабораторной работы № 1 для заданных значений диапазона и шага аргументов z, t, таблица 5. Первое значение вычисляется в крайней левой точке заданного диапазона значений аргументов x, t. Произвести тестирование программы.

Таблица 5

Вариант	Диапазон значений аргумента	Шаг аргумента	Вариант	Диапазон значений аргумента	Шаг аргумента
1	[0,1, 2,0]	0,2	2	[0,3, 2,0]	0,25
3	[1,5, 2,5]	0,05	4	[0,8, 1,3]	0,04
5	[–0,5, 0,5]	0,1	6	[6,0, 6,5]	0,05
7	[1, 2]	0,11	8	[–3, 3]	0,5
9	[2,0, 2,5]	0,05	10	[1,5, 3,5]	0,2
11	[–6, –5]	0,08	12	[0, 10]	0,6
13	[–3, 0,5]	0,45	14	[0,1, 0,5]	0,03
15	[10, 12]	0,11	16	[3,5, 5,6[	0,11
17	[0,5, 0,9[	0,03	18	[2, 4[	0,13
19	[8,0, 9,4[	0,11	20	[3, 5[	0,14
21	[0, 1[	0,105	22	[2, 3[	0,07
23	[0, 2,2[	0,12	24	[–4, 3[	0,9
25	[0,5, 1[	0,04	26	[1,8, 2,1[	0,023
27	[0,1, 0,3[	0,01	28	[3, 4[	0,08
29	[1,5, 2[	0,081	30	[0, 0,5[	0,035

СОДЕРЖАНИЕ ОТЧЕТА

1. Лабораторное задание.
2. ССА.
3. Текст программы.

Лабораторная работа № 4

ЦИКЛ С ПОСТУСЛОВИЕМ

ЦЕЛЬ РАБОТЫ: Изучить цикл с постусловием; познакомиться с процессом отладки программ.

Алгоритмизация. В настоящей работе рассматривается оператор REPEAT … UNTIL, реализующий алгоритмическую структуру «цикл-до», рисунок 7а. Эта алгоритмическая структура не является базовой, но – дополнительной. Рисунки 7б, 7в можно понять по аналогии с рисунками 5б, 5в.

а) б) в)

Рисунок 7

Программирование. Цикл с постусловием имеет следующие отличия от цикла с предусловием. Во-первых, в цикле с постусловием проверка условия продолжения цикла осуществляется *после* итерации, поэтому цикл может совершить хотя бы одну итерацию. Во-вторых, цикл While продолжает выполнение, когда условие истинно, а цикл Repeat – когда условие ложно. Часто их путают, но понять и запомнить очень легко. *Условие на входе – это условие входа* в цикл и продолжения его работы; *условие на выходе – это условие выхода*. Т. о., когда выполняется условие входа, осуществляется вход в цикл или продолжение его выполнения; когда выполняется условие выхода – осуществляется выход из цикла.

Оператор цикла с постусловием имеет следующую структуру:

REPEAT *тело цикла* UNTIL *условие*,

где *тело цикла* – произвольная последовательность операторов Турбо Паскаля.

Пример 4.1. Написать алгоритм и программу вычисления функции y(x) из Примера 2.1 в N=10 равномерно распределенных в диапазоне X1≤x≤X2 точках, где X1= –1; X2=2. Результаты сформировать в виде таблицы.

Схема алгоритма изображена на рисунке 8.

```
Program lab4;
Const X1=-1; X2=2; N=10;
Var y,x,dx: Real;
    i:Integer;
```

```
Begin
   WriteLn('!---!-----!-----!');
   WriteLn('! n !  x  !  y  !');
   WriteLn('!---!-----!-----!');
   dx:=(X2-X1)/(N-1);
   x:=X1;
   i:=1;
   Repeat
      If x<=0 Then y:=(sin(x)/cos(x)+ln(abs(x-1))/ln(10))/(2*x+5)
              Else If x<=1 Then y:=exp(ln(x)/3)
                           Else y:=5*sin(2*x);
      WriteLn(i:3,x:7:2,y:6:2);
      x:=x+dx;
      i:=i+1;
   Until  x>2;
End.
```

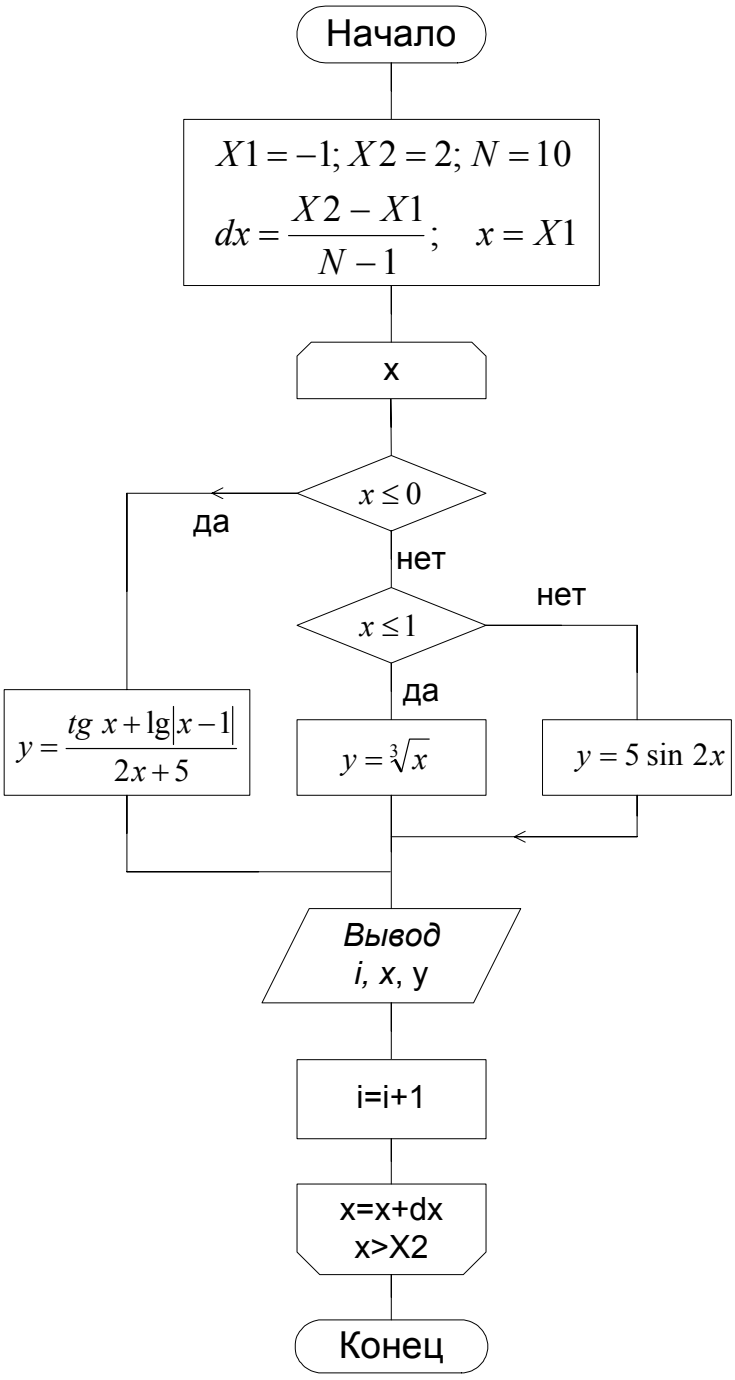


Рисунок 8

Отладка программ. Различают два вида ошибок программирования: синтаксические и логические. Подобно тому, как грамматически безупречный текст может быть полнейшей бессмыслицей, программа, не имеющая синтаксических ошибок, может содержать логические ошибки. Компилятор выявляет только синтаксические ошибки. Увеличение сложности программ неизбежно приводит к логическим ошибкам, которые выявляют в процессе отладки.

В процессе отладки можно по шагам выполнять программу, наблюдая за переменными. Выбор переменных для наблюдения осуществляется через меню **Debug → Add watch…** или комбинацией клавиш Ctrl+F7. Выполнение одного шага програм-

мы осуществляется через меню **Run → Trace into** или клавишей F7. Прервать процесс отладки можно через меню **Run → Program reset** или комбинацией Ctrl+F2. Во время отладки можно заглядывать в окно вывода, нажимая Alt+F5, либо через меню **Debug → Output** сделать это окно постоянно присутствующим на экране. Менять размеры окон можно, захватывая левой кнопкой мыши за нижний правый угол окна, а перемещать окна по экрану, захватывая мышкой за верхнюю границу окна. Закрыть окно можно, щелкнув мышью по желтому прямоугольнику в правом верхнем углу окна.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Задание 1. Написать алгоритм и программу вычисления функции y(x) из лабораторной работы № 2 в N равномерно распределенных в диапазоне $X1 \le x \le X2$ точках, таблица 6. Результаты сформировать в виде таблицы. Произвести тестирование.

Таблица 6

Вариант	X1	X2	N	Вариант	X1	X2	N
1	–1	2	10	2	1	4	12
3	0	2	15	4	–2	0	14
5	0,1	1,8	16	6	–0,5	1	18
7	1	7	20	8	–1	2	11
9	–1	1	9	10	0	1	8
11	1	3	7	12	–2	–1	6
13	–0,1	0,3	5	14	1	100	20
15	–2	2	19	16	1	3	18
17	10	30	17	18	–5	5	16
19	–2	4	15	20	0,1	1,9	14
21	1,5	3,5	13	22	0,2	0,6	12
23	0,2	0,6	11	24	2	4	10
25	2,5	4,4	9	26	–1	1	7
27	2,5	7,5	5	28	8	10	7
29	–2	0	9	30	–1	1	11

Задание 2. Произвести пошаговое выполнение программы, наблюдая переменные x, y, i.

СОДЕРЖАНИЕ ОТЧЕТА

1. Лабораторное задание.
2. Структурная схема алгоритма.
3. Текст программы.

Лабораторная работа № 5

ЦИКЛ С ЗАДАННЫМ ЧИСЛОМ ПОВТОРЕНИЙ

ЦЕЛЬ РАБОТЫ: Изучить цикл с заданным числом повторений (счетный цикл).

Алгоритмизация. В настоящей работе рассматривается оператор FOR, реализующий алгоритмическую структуру «цикл с заданным числом повторений», рисунок 9. Данный цикл работает следующим образом. В начале работы цикла целочисленный параметр i принимает значение n1 и цикл совершает первую итерацию. Если i≠n2, параметр автоматически изменяется на единицу и цикл совершает следующую итерацию. Если i=n2, цикл совершает последнюю итерацию. В случае, если n1=n2, цикл совершит первую и последнюю (одну) итерацию. Таким образом, число повторений равно | n2–n1| +1. Знак модуля в этой формуле необходим для того, чтобы формула сохранила общность для двух форм счетного цикла. В первой форме счетного цикла n2>n1 и параметр i увеличивается на единицу. Во второй форме счетного цикла n2<n1 и параметр i уменьшается на единицу.

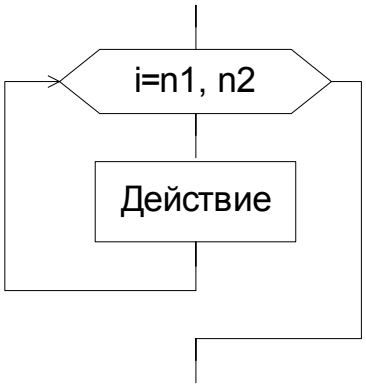


Рисунок 9

Программирование. В языке Паскаль первая форма счетного цикла имеет структуру:

FOR *параметр_цикла*:=n1 TO n2 DO *оператор*;

вторая форма счетного цикла выглядит следующим образом:

FOR *параметр_цикла*:=n1 DOWNTO n2 DO *оператор*,

где *параметр_цикла* – целочисленная переменная, соответствует i на рисунке 9; n1, n2 – целочисленные переменные или константы; *оператор* – произвольный оператор Турбо Паскаля, соответствует элементу «действие» на рисунке 9.

Пример 5.1. Изменить алгоритм и программу из лабораторной работы 4, используя вместо цикла с постусловием счетный цикл.

Для решения данной задачи разработаны структурная схема алгоритма (рисунок 10) и программа lab5_1.

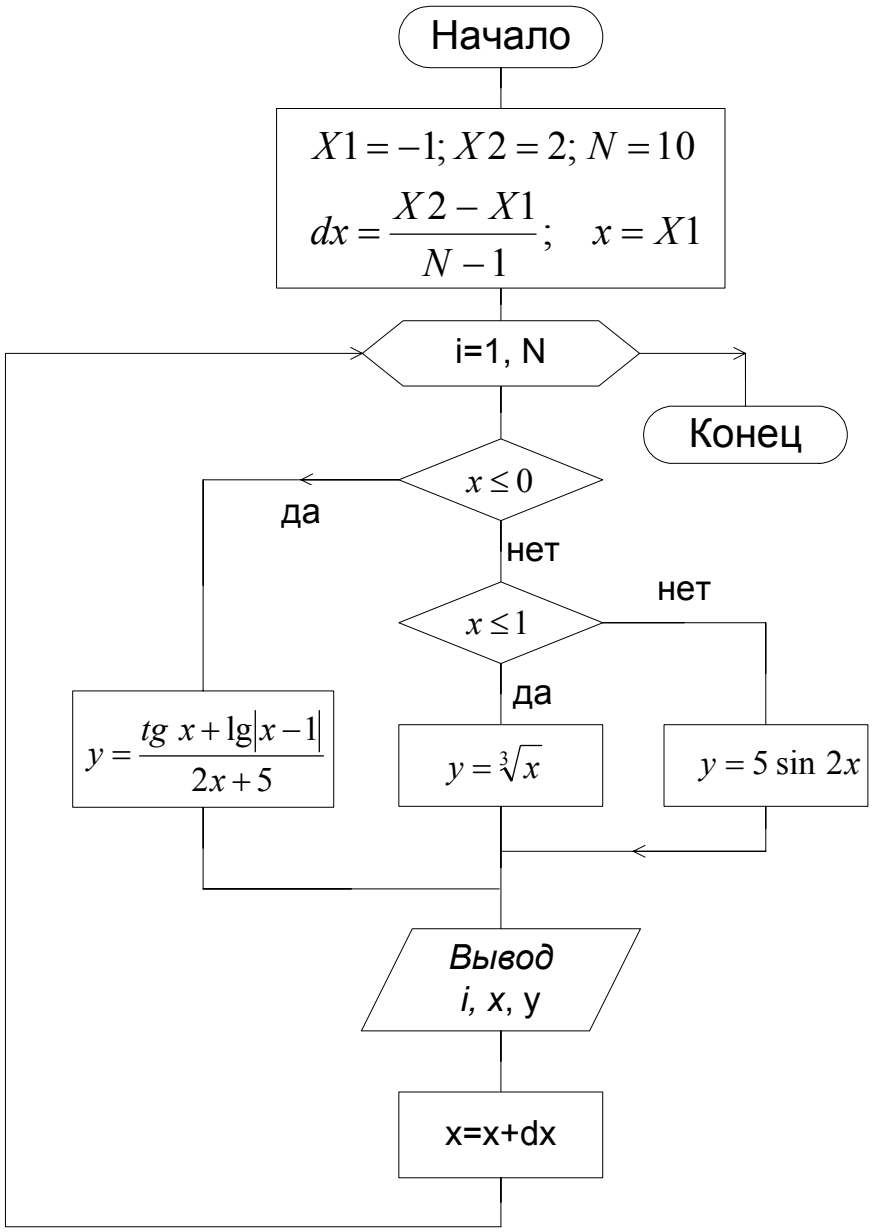


Рисунок 10

```
Program lab5_1;
Const X1=-1; X2=2; N=10;
Var y,x,dx: Real;
    i:Integer;
Begin
   WriteLn('!---!-----!-----!');
   WriteLn('! n !  x  !  y  !');
   WriteLn('!---!-----!-----!');
   dx:=(X2-X1)/(N-1);
   x:=X1;
   For i:=1 to N do
      Begin
         If x<=0 Then y:=(sin(x)/cos(x)+ln(abs(x-1))/ln(10))/(2*x+5)
                 Else If x<=1 Then y:=exp(ln(x)/3)
                              Else y:=5*sin(2*x);
```

```
        WriteLn(i:3,x:7:2,y:6:2);
        x:=x+dx;
      End
End.
```

Пример 5.2. Для заданного $0<x<1$ и заданного числа слагаемых вычислить сумму

$$s = x - \frac{1}{2}\cdot\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\cdot\frac{x^5}{5} - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot\frac{x^7}{7} + \ldots$$

Р е ш е н и е.

Если число слагаемых более одного, искомая сумма s определяется формулой:

$$s := x + \sum_{i=1}^{n}\left[(-1)^i\cdot\left[\prod_{k=1}^{i}\left[\frac{(2\cdot k-1)}{(2\cdot k)}\right]\right]\cdot\frac{x^{2\cdot i+1}}{2\cdot i+1}\right].$$

Число слагаемых в искомой сумме равно n+1.

Для решения данной задачи разработаны: структурная схема алгоритма, рисунок 11, и программа lab5_2.

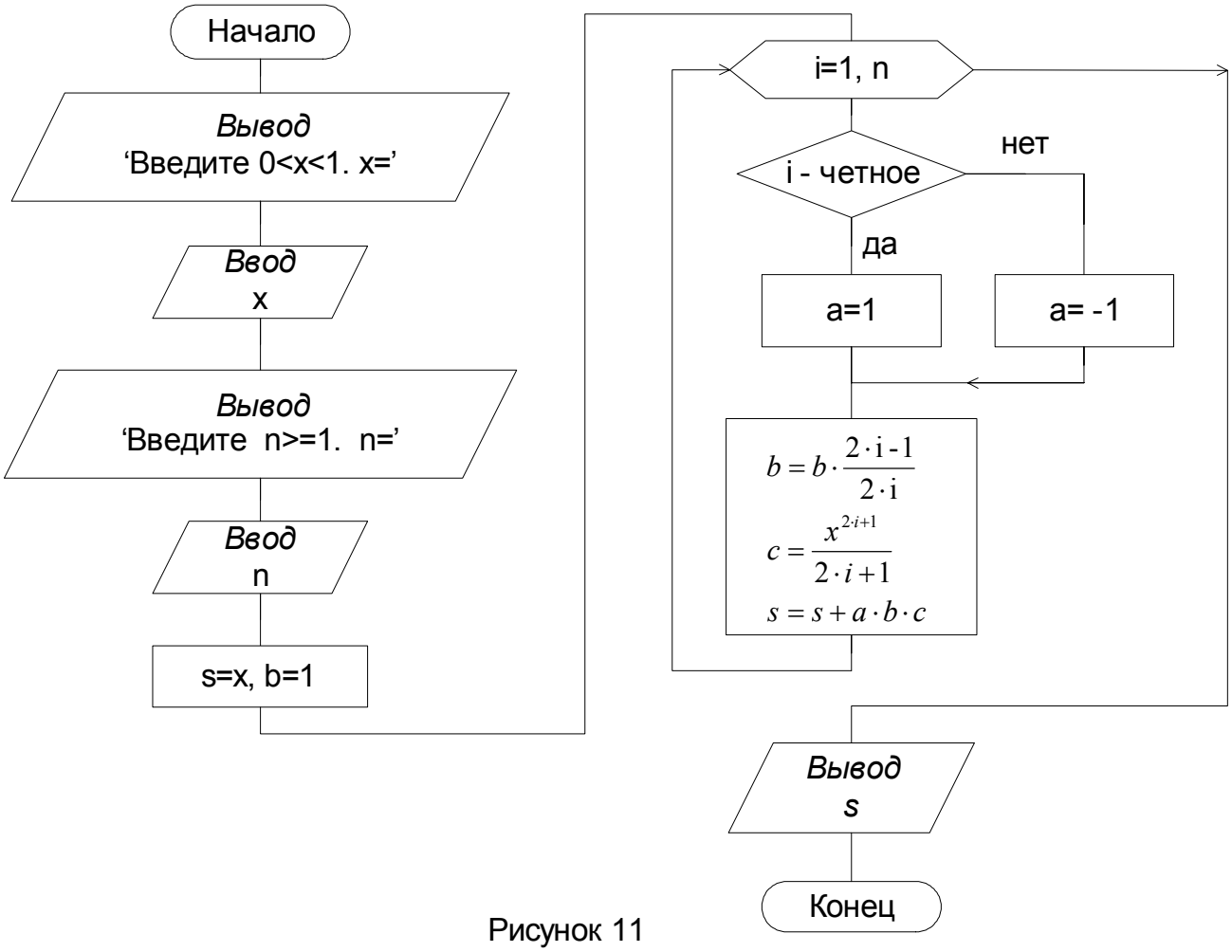


Рисунок 11

```
Program lab5_2;
Var a, b, c, x, s: Real;
    n,i: Integer;
Begin
   Write('Введите 0<x<1.  x=');
   Read(x);
   Write('Введите n>=1.  n=');
   Read(n);
   s:=x;  b:=1;
   For i:=1 to n do
      begin
        If odd(i) Then a:=-1
                  Else a:=1;
        b:=b*(2*i-1)/(2*i);
        c:=exp((2*i+1)*ln(x))/(2*i+1);
        s:=s+a*b*c;
      end;
   WriteLn('Summa=', s:8:5);
End.
```

Комментарии к программе. Отрицательное число (–1) нельзя возвести в произвольную степень, пользуясь рассмотренной конструкцией a^i =exp(i*ln(a)), поскольку отрицательное число под знаком логарифма приведет к ошибке. Вместо этого выбирается множитель единица либо минус единица, в зависимости от четности i. Если i – нечетное, то функция odd(i) возвращает значение ИСТИНА.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Задание 1. Изменить алгоритм и программу из лабораторной работы 4, используя вместо цикла с постусловием счетный цикл.

Задание 2. Для заданного *x*, числа слагаемых или *n* вычислить следующее.

1. $s = \frac{x}{1!} + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots + \frac{x^{2n+1}}{(2n+1)!}$.
2. $s = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots + \frac{x^{2n}}{(2n)!}$.
3. $s = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, \quad x > -1$.
4. $s = -\frac{x}{1} - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} + \ldots, \quad x < 1$.
5. $s = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

6. $s = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$

7. $s = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$

8. $s = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

9. $s = 2 \cdot \left[x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots + \frac{x^{2n+1}}{(2n+1)} \right], \quad x < 1.$

10. $s = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \ldots, \quad x < 1.$

11. $s = 1 - \frac{x^2}{1!} + \frac{x^4}{2!} - \ldots + (-1)^n \cdot \frac{x^{2n}}{n!}.$

12. $s = x - \frac{1}{1!}\frac{x^3}{3} + \frac{1}{2!}\frac{x^5}{5} - \ldots + (-1)^n \cdot \frac{1}{n!} \cdot \frac{x^{2n+1}}{2n+1}.$

13. $s = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \ldots$

14. $s = x - \frac{x^3}{3 \cdot 3!} + \frac{x^5}{5 \cdot 5!} - \frac{x^7}{7 \cdot 7!} + \ldots$

15. $s = 1 - \frac{2 \cdot 3}{2}x + \frac{3 \cdot 4}{2}x^2 - \frac{4 \cdot 5}{2}x^3 + \ldots$

16. $s = 1 + \frac{1}{2}x - \frac{1}{2 \cdot 4}x^2 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6}x^3 - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 8}x^4 + \ldots$

17. $s = 1 - \frac{1}{2}x + \frac{1 \cdot 3}{2 \cdot 4}x^2 - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6}x^3 + \frac{1 \cdot 3 \cdot 5 \cdot 7}{2 \cdot 4 \cdot 6 \cdot 8}x^4 - \ldots$

18. $s = 1 + \frac{1}{2}x^2 + \frac{1 \cdot 3}{2 \cdot 4}x^4 + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6}x^6 + \frac{1 \cdot 3 \cdot 5 \cdot 7}{2 \cdot 4 \cdot 6 \cdot 8}x^8 + \ldots$

19. $s = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \ldots$

20. $s = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + \frac{(-1)^n}{(2n+1)} \right).$

21. $Z = \prod_{i=2}^{10} (x + i)/i.$

22. Вычислить сумму четных и сумму нечетных чисел натурального ряда от 1 до n.

23. Вычислить сумму положительных и сумму отрицательных чисел ряда $(-1)^i \frac{x}{i}$ для i=1…n.

24. $s(x) = 2\left[\frac{\sin x}{1} - \frac{\sin 2x}{2} + \frac{\sin 3x}{3} - \ldots + (-1)^{n+1}\frac{\sin nx}{n}\right]$.

25. $s(x) = \frac{\pi}{2} - \frac{4}{\pi}\left(\frac{\cos x}{1^2} + \frac{\cos 3x}{3^2} + \ldots + \frac{\cos(2n-1)x}{(2n-1)^2}\right)$.

26. $s(x) = \frac{\pi^2}{3} - 4\left(\frac{\cos x}{1^2} - \frac{\cos 2x}{2^2} + \frac{\cos 3x}{3^2} - \frac{\cos 4x}{4^2} + \ldots\right)$

27. $s(x) = \sin x + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \ldots + \frac{\sin(2n-1)x}{2n-1}$.

28. Для заданных A, t, n, q, ω вычислить $s(t) = \frac{A}{q}\left[1 + 2\sum_{k=1}^{n}\frac{\sin(k\pi/q)}{k\pi/q}\cdot\cos k\omega t\right]$.

29. $s(x) = \frac{x}{3} - \frac{1}{7}\cdot\frac{x^3}{3!} + \frac{1}{11}\cdot\frac{x^5}{5!} \mp \ldots$

30. $s(x) = 1 - \frac{1}{5}\cdot\frac{x^2}{2!} + \frac{1}{9}\cdot\frac{x^4}{4!} - \frac{1}{13}\cdot\frac{x^6}{6!} \mp \ldots$

СОДЕРЖАНИЕ ОТЧЕТА

1. Лабораторное задание.
2. Структурная схема алгоритма.
3. Текст программы.

Лабораторная работа № 6

ФАЙЛЫ, ОДНОМЕРНЫЕ МАССИВЫ

ЦЕЛЬ РАБОТЫ: научиться работать с одномерными массивами и файлами.

Массивы. Массивы принадлежат к *структурированным типам данных*, которые, в отличие от простых типов (напр.: integer, real), содержат несколько компонентов. Причем, у массивов все компоненты – одного типа. Обращение к отдельным компонентам осуществляется посредством имени массива и индексом ячейки в прямоугольных скобках. Например, *a*[8] – ячейка 8 массива *a*. Индекс может быть цифровой, буквенный и вообще – перечисляемым типом.

Массивы определяют в разделах описания типов или переменных следующим образом:

TYPE Имя_типа = ARRAY [диапазон_индексов] OF базовый_тип;

VAR Имя_переменной : ARRAY [диапазон_индексов] OF базовый_тип;

Диапазон указывает значения индексов первого и последнего элементов массива.

Примеры:

```
Var a: array [1..10] of Real;
    b, c: array [-15..20] of Char;
    d: array ['a'..'z'] of Integer;
    e: array [Shortint] of array [3..5] of Boolean;
```

Последняя строка определяет массив из 256 элементов, каждый из которых является массивом из трех элементов булевского типа.

Выше приведены примеры, где массивы описаны в разделе описания переменных. Но можно описывать переменные типа массив (через тип):

```
Type mas=array[1..10] of Real;
Var a,b: mas;
```

В последнем случае в теле программы возможно присваивание `a:=b`, чего нельзя сделать, если массивы описаны непосредственно в разделе описания переменных.

Файлы. Ввод из файла и вывод в файл осуществляется соответственно процедурами Read(), ReadLn() и Write(), WriteLn(). В аргументе этих процедур указывают, в какой файл нужно осуществлять вывод или ввод. По умолчанию (как мы делали до сих пор) – в стандартный файл вывода (дисплей) и из стандартного файла ввода (клавиатура). В аргументе этих процедур указывают, однако, не имя файла, а переменную файлового типа, связанную с именем требуемого файла:

Read(файловая_переменная, список_вводимых_переменных);

Write(файловая_переменная, список_выводимых_переменных);

С учетом этого, для работы с файлами имеем следующий алгоритм работы.

1. Выполнить процедуру ASSIGN(<ф. п.>, <имя файла>), которая связывает файловую переменную <ф. п.>, объявленную в разделе var, с именем файла.

2. Если требуется читать из файла, то процедурой RESET(<ф. п.>) открываем файл для чтения. Если требуется записывать в файл, то процедурой REWRITE(<ф. п.>) открываем файл для записи.

3. Вывод в файл или ввод из него осуществляется процедурами WRITE(<ф.п.>, …) и READ(<ф.п.>, …) соответственно. Первым параметром этих процедур является переменная файлового типа.

4. По окончании работы с файлом закрываем файл процедурой CLOSE(<ф.п.>).

Пример 6.1. Дан одномерный массив чисел, записанный в текстовый файл. Найти максимальное число и записать его вместе с исходным массивом в другой текстовый файл. Размер массива задан первым числом.

В ходе решения данной задачи были разработаны структурная схема алгоритма (рисунок 12) и программа `lab6_1`.

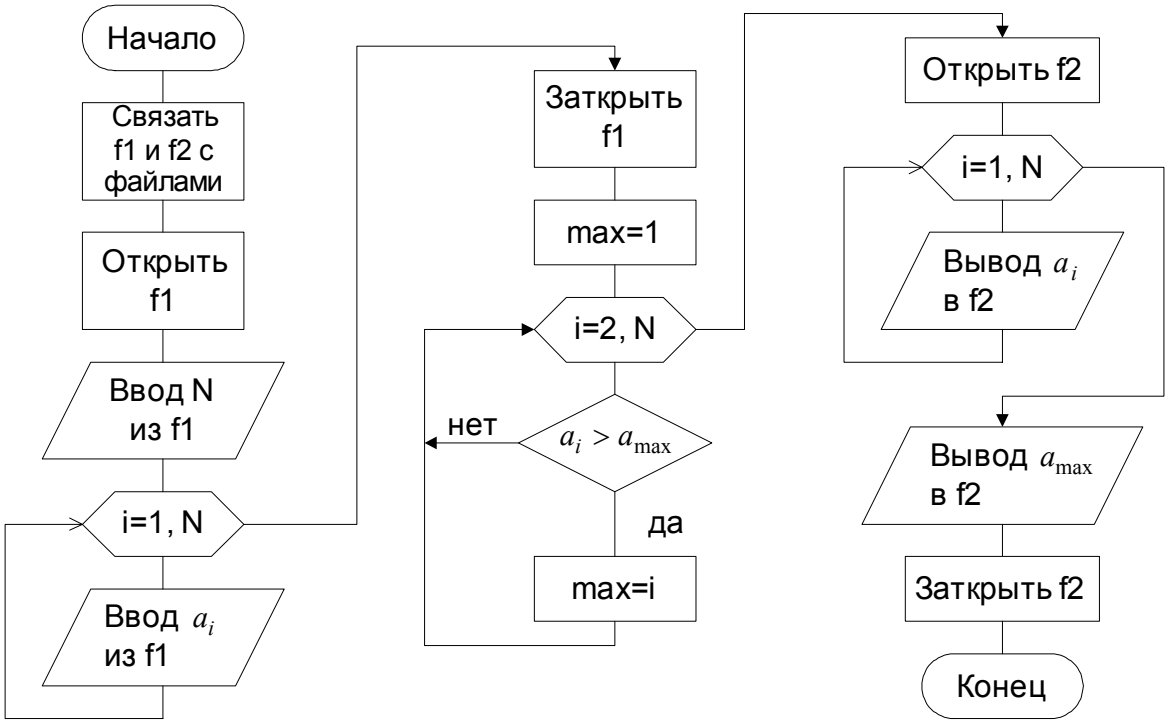


Рисунок 12

```
Program lab6_1;
Var i, max, N: Integer;
    a: array [1..100] of integer;
    f1, f2: Text; {Файловые переменные}
Begin
   assign(f1, 'in.txt');
   assign(f2, 'out.txt');
   Reset(f1);
   Read(f1, N);
   For i:=1 to N do Read(f1, a[i]);
   Close(f1);
   max:=1;  {Номер максимального элемента}
   For i:=1 to N do If a[max]<a[i] Then max:=i;
   Rewrite(f2);
   WriteLn(f2,'Original array');
   For i:=1 to N do Write(f2,a[i],' ');
   WriteLn(f2);
   WriteLn(f2,'Maximal Value=',a[max]);
   Close(f2)
End.
```

В качестве теста был создан файл `in.txt`, имеющий следующее содержание:

```
12 0 -5 43 199 4 -9 13 14 15 16 17 -200
```

В результате выполнения программы был получен файл `out.txt`:

```
Original array
```

```
0 -5 43 199 4 -9 13 14 15 16 17 -200
Maximal Value=199
```

Заметим, что в процедурах ASSIGN имена файлов указаны без путей. Поэтому входной файл должен быть помещен в ту же директорию, что и программа lab6_1, и выходной файл будет создан в той же директории.

Обратите внимание: ввод и вывод данных структурированных типов (например, массивов) осуществляется поэлементно, а не одновременно всем массивом.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Общее условие. Написать алгоритм и программу, которые исходный числовой массив данных (если он есть) считывают из текстового файла f1, а выходную информацию записывают в текстовый файл f2. Произвести тестирование программы.

Условия по вариантам.

1. Для N чисел найти математическое ожидание (среднее арифметическое) и дисперсию по следующим формулам.

$$M = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad D = \sqrt{\sum_{i=1}^{N} (x_i - M)^2 / (N-1)}.$$

2. Для N чисел найти среднее геометрическое

$$G = \sqrt[N]{\prod_{i=1}^{N} x_i}.$$

3. Найти сумму максимального и минимального элементов массива.

4. Найти второй (по модулю) элемент массива.

5. Если таковые имеются, вывести вначале все положительные, затем – все отрицательные, затем – все нулевые элементы исходного массива.

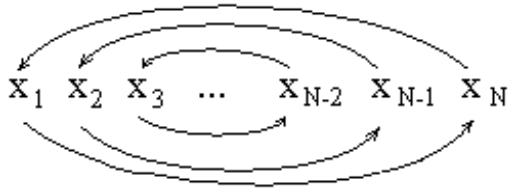
6. Для массива X из N элементов произвести циклическую перестановку следующим образом: $x_{i+1} \rightarrow x_i$, $x_1 \rightarrow x_N$.

7. В исходном массиве поменять местами первый нулевой элемент и первый элемент с максимальным модулем.

8. Дан массив A из N элементов. Сформировать массив B такой же размерности по правилу:

$b_i = \sin(a_i)$, если N – нечетное,
$b_i = \cos(a_i)$, если N – четное.

9. Дан массив X с четным числом элементов. Произвести перестановку элементов по следующей схеме.



10. Даны два массива A и B по 10 элементов. Сформировать массив C по правилу: $c_i=a_i+b_i$, если a_i и b_i различны; иначе $c_i = a_i$.

11. Дан массив из N элементов. Найти порядковый номер элемента наиболее близкого к числу, введенному с клавиатуры.

12. Сформировать массив из 10 простых чисел.

13. Дан массив из 20 элементов. Из исходного массива сформировать другой массив заменой нулевых элементов (если они есть) значениями, равными максимальному элементу.

14. Дан массив из 20 элементов. Сформировать массив, составленный из N первых элементов исходного массива, где N – модуль разности порядковых номеров максимального и минимального элементов исходного массива.

15. Дан массив из 15 элементов. Сформировать новый массив, составленный и элементов исходного массива за исключением минимального и максимального элементов.

16. Дан массив A из N элементов. Сформировать массив B, элементы которого равны: $b_1=a_1$, $b_2=a_1+a_2$, $b_3=a_1+a_2+a_3$ и т. д.

17. Из исходного массива целых чисел сформировать массив, элементами которого являются K последних элементов исходного массива, где K – число отрицательных элементов исходного массива.

18. Дан массив из N элементов. Из исходного массива сформировать два массива: первый – из элементов с четными, а второй – с нечетными значениями.

19. Дан массив A из четного числа N элементов. Сформировать массив B из K=N/2 элементов так, чтобы $b_1=a_N+a_{N-1}$, $b_1=a_{N-2}+a_{N-3}$, … , $b_{K-1}=a_4+a_3$, $b_K=a_2+a_1$.

20. Сформировать массив из таких элементов исходного массива, что эти элементы имеют:

- четные значения, если сумма элементов с четными значениями исходного массива больше суммы элементов с нечетными значениями;
- нечетные значения, если сумма элементов с четными значениями исходного массива меньше суммы элементов с нечетными значениями;
- нулевые значения (если они есть), если сумма элементов с четными значениями исходного массива равна сумме элементов с нечетными значениями.

21. Сформировать массив, в который должны входить:

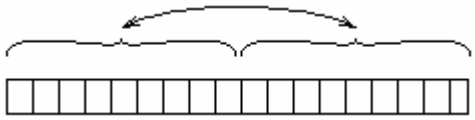
- элементы исходного массива с четными номерами, если сумма элементов с нечетными номерами больше суммы элементов с четными номерами;
- элементы исходного массива с нечетными номерами, если сумма элементов с нечетными номерами не больше суммы элементов с четными номерами.

22. Из исходного массива сформировать массив, содержащий:

- элементы исходного массива, переписанные в обратном порядке, если среднее арифметическое элементов больше 10;
- элементы исходного массива, записанные в прямом порядке, если среднее арифметическое элементов не больше 10.

23. Дан массив X с четным количеством элементов. Если сумма элементов четная, то осуществить перестановку четных и нечетных элементов по следующей схеме: $x_1 \leftrightarrow x_2$; $x_3 \leftrightarrow x_4$; …; $x_{N-1} \leftrightarrow x_N$. Иначе – не переставлять.

24. Если сумма элементов исходного массива больше нуля, то поменять местами первую и вторую половины исходного массива; иначе – не менять.



25. Сформировать массив из всех элементов исходного массива, которые больше или равны среднему арифметическому значению элементов исходного массива.

26. Произвести перестановку элементов исходного массива так, чтобы отрицательные и неотрицательные элементы чередовались. Например:

исходный массив: 12 0 –3 5 –16 –7 9 10 4;

выходной массив: 12 –3 0 –16 5 –7 9 10 4.

27. Найти среднее арифметическое SA элементов исходного массива. Составить новый массив из элементов с четными значениями, если SA, округленное до ближайшего целого – четное значение; иначе – из элементов с нечетными значениями.

28. Сформировать массив, составленный из элементов исходного массива, расположенных:

- между первым и последним элементами, имеющими нулевое значение, если нулей больше одного;
- между нулем и последним элементом, если нуль один;
- между первым и последним элементами, если нулей нет.

29. Сформировать массив, составленный из элементов исходного массива:

- начиная первым элементом, имеющим отрицательное значение, и кончая последним элементом, если отрицательные элементы имеются;
- начиная первым и кончая последним элементом, если отрицательные элементы отсутствуют.

30. Сформировать массив, составленный из элементов исходного массива. В результирующий массив входят элементы;

- начиная первым элементом и кончая последним, имеющими четные значения, если таких элементов более одного;
- начиная первым элементом и кончая элементом, имеющим четное значение, если четный элемент один.

Если элементы, имеющие четные значения, отсутствуют, то выходной массив не должен содержать ни одного элемента.

Лабораторная работа № 7

ПРОЦЕДУРЫ И ФУНКЦИИ

ЦЕЛЬ РАБОТЫ: научиться разрабатывать алгоритмы и программы с использованием процедур и функций.

Алгоритмизация

При возрастании сложности задач, рационально разбивать сложную задачу на ряд мелких подзадач, каждую из которых гораздо легче решить в отдельности. С помощью подпрограмм большая программа (алгоритм) может быть разбита на не-

сколько относительно крупных и в значительной степени независимых частей. Если требуется, полученные части разбиваются далее на более мелкие и т.д. до тех пор, пока они станут настолько простыми, чтобы их можно было легко запрограммировать. Такой способ отвечает современным методам нисходящего проектирования программ.

Таким образом, единая схема алгоритма распадается на две или несколько завершенных частей: общую часть алгоритма, где подпрограммы лишь упоминаются (откуда *вызываются*), и другие части, каждая из которых раскрывает алгоритм работы одной из подпрограмм. В месте вызова подпрограммы помещают элемент «предопределенный процесс», см. приложение 1. Внутри элемента пишут имя подпрограммы и в скобках указывают список аргументов (при наличии аргументов), рисунок 13а. ССА подпрограммы, как и ССА программы, помимо прочих элементов имеет начальный и конечный терминаторы. Разница лишь в том, что начальный терминатор подпрограммы вместо слова «Начало» содержит имя подпрограммы со списком аргументов в скобках (при наличии аргументов), а конечный терминатор вместо слова «Конец» содержит фразу «Конец п/п», рисунок 13б.

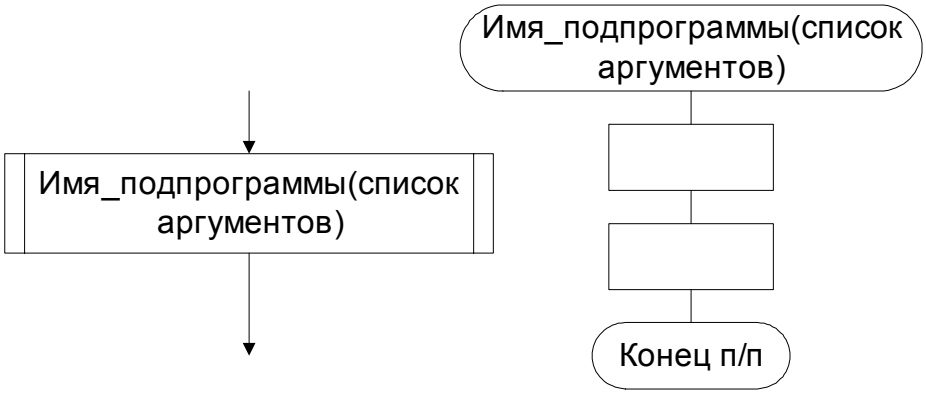


Рисунок 13

Программирование

Отличие процедур от функций, их применение. И функции и процедуры являются подпрограммами. Чем процедура отличается от функции? Тем, что функция *возвращает значение*, а процедура не возвращает; поэтому процедуры используют, как самостоятельные операторы, а функции – как составные части операторов.

Например, пусть функция sin(x) находится в составе оператора присваивания

```
y:= 2*(sin(x)-1);
```

или в составе оператора вывода результатов

```
WriteLn('sin(x)=', sin(x));
```

Вместо sin(x) вернется значение (число), вычисленное по формуле

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Это значение попадет в первом примере опять в формулу и результат новых вычислений будет присвоен переменной y. Во втором примере sin(x) вернет численное значение в аргумент процедуры WriteLn(), которое будет выведено на экран вслед за текстовой константой `'sin(x)='`.

Попытка использовать функцию в качестве самостоятельного оператора

```
Sin(x);
```
приведет к ошибке. И в самом деле, что же нужно сделать с вычисленным значением? куда его девать?

Процедура же, напротив, используется как самостоятельный оператор (например, процедура WriteLn()). Попытка включить процедуру, скажем, в состав формулы в операторе присваивания

```
y:=WriteLn(x)+1;
```
приведет к ошибке, поскольку процедура не возвращает числа, которое требуется для подстановки в формулу.

Описание процедур и функций, их параметры. Глобальные и локальные переменные. Описания процедур и функций помещают после раздела описания переменных программы и перед телом программы.

Описание подпрограммы состоит из следующих частей:

1) заголовок подпрограммы;

2) раздел описания собственных переменных и констант (если они имеются);

3) тело подпрограммы, ограниченное операторными скобками.

Заголовки подпрограмм. Заголовок процедуры имеет следующий вид:

PROCEDURE <имя> [(<список формальных параметров>)];

заголовок функции:

FUNCTION <имя> [(<список формальных параметров>)]: <тип>;

где <имя> – имя подпрограммы; <тип> – тип возвращаемого функцией результата.

В списке формальных параметров (если он есть) перечисляются переменные: однотипные – через запятую; разнотипные – через точку с запятой. Примеры нескольких заголовков подпрограмм:

Function Cos(x: real): real; – функция с именем «Cos», имеющая параметр типа real и возвращающая значение того же типа;

Function F1(x,y: real): integer; – функция F1 с двумя параметрами типа real, возвращающая целое число;

Function Ksi: real; – функция Ksi без параметров, возвращающая действительное число;

Procedure Pr1; – процедура Pr1 без параметров;

Procedure Pr2(a : integer); – процедура с одним целочисленным параметром;

Procedure Pr3(a, b: real; d, e, f : integer; g: string); – процедура с двумя параметрами типа real; тремя параметрами типа integer и одним – типа string.

Раздел описания переменных подпрограммы. Если для промежуточных вычислений внутри подпрограммы необходимо ввести переменные или константы, которые не будут использованы вне подпрограммы, после заголовка подпрограммы помещают разделы описания переменных и констант подпрограммы:

```
PROCEDURE <имя> [(<список формальных параметров>)];
Const  {локальные константы}
Var {локальные переменные}
BEGIN
{ операторы подпрограммы}
```

```
END;
```

Такие переменные и константы существуют только во время работы подпрограммы, а область их действия – тело подпрограммы. Поэтому такие переменные и константы называют *локальными*.

Внутри подпрограмм можно использовать и переменные, описанные в основной программе. Эти переменные существуют, пока работает программа; область их действия – вся программа. Такие переменные и константы называют *глобальными*.

Обмен данными между подпрограммой и вызывающей программой.

Пусть функция имеет следующий заголовок:

```
Function Sin(x:real):real;
```

Здесь передача данных ***в*** подпрограмму осуществляется через ее параметр `x`, а ***из*** подпрограммы – путем возвращения функцией значения.

Как было сказано выше, локальные переменные существуют только во время работы подпрограммы, поэтому подпрограмма не может оставить в них результаты своей работы. Глобальные же переменные существуют до, во время и после работы подпрограммы. Поэтому через них можно передать данные как ***в*** подпрограмму, так и ***из*** нее.

Существует две разновидности обмена данными через глобальные переменные.

Первая – обращаться в подпрограмме к глобальным переменным непосредственно через их имена.

Вторая – через имена формальных параметров. Однако для этого, в разделе описания перед именем формального параметра должно стоять слово «var». Такой формальный параметр называют *параметр-переменная*. Формальные параметры, перед которыми не стоит слово «var», называют *параметры-значения*. Например, в процедуре

```
Procedure Proc1(a, b: real; var c, d: integer );
```

`a`, `b` – параметры-значения; а `c`, `d` – параметры-переменные. При этом тип формального параметра-переменной должен совпадать с типом *фактического параметра* (т. е. параметра, который стоит на месте формального при вызове подпрограммы). При обращении в подпрограмме к имени параметра-переменной мы обращаемся к фактическому параметру, т. е. к глобальной переменной. Под именем же параметра-значения, скрывается вычисленное значение выражения, находящегося на месте формального параметра *в момент вызова* подпрограммы. Вычисленное значение *копируется* в ячейку памяти и передается в подпрограмму. Т. о., после вызова подпрограммы формальный параметр-значение и фактический параметр никак не связаны, даже если фактическим параметром является выражение, состоящее из одного лишь имени глобальной переменной, поскольку подпрограмма в этом случае имеет дело с копией фактического параметра.

Для пояснения изложенных выше теоретических сведений продемонстрируем вычисление суммы $\frac{x}{1}+\frac{x}{2}+\ldots+\frac{x}{20}$ тремя различными способами: с помощью одной функции, а затем – с помощью двух процедур.

Ниже приведена программа, использующая функцию.

```
Program Lab7_01;
var x: real;       {Раздел описания переменных программы.
(Глобальные переменные)}

{---Начало описания функции F1---}
Function F1(k:Real):Real;      {Заголовок функции}
Var i: Integer;          {Локальные переменные}
    s: Real;
Begin    {of F1    (Начало тела функции)}
  s:=0;
  For i:=1 to 20 do s:=s+k/i;
  F1:=s;
End;     {of F1    (Конец тела функции)}
{---Конец описания функции F1---}

Begin {Of Program (Начало тела программы)}
   x:=1;
   WriteLn('F1(x)=', F1(x):4:2);
End. {Of Program (Конец тела программы)}
```

В результате выполнения программы мы получим на экране следующее:

```
F1(x)=3.60
```

К о м м е н т а р и и к программе Lab7_01.

1. Как видно из текста, в программе Lab7_01 список формальных параметров функции F1 состоит из единственного параметра-значения типа real. Возвращает функция число такого же типа.

2. В теле программы мы видим, что функция помещена в аргумент оператора вывода на экран, куда она и вернет вычисленное значение.

3. Фактическим параметром является выражение, состоящее из одного имени (x).

4. Чтобы функция смогла вернуть требуемое значение, в тело функции необходимо поместить оператор, который присваивает имени функции результат вычислений (F1:=s).

Далее приведена программа, решающая ту же задачу, но производящая обмен через глобальные переменные.

```
Program Lab7_02;
var x,y,z: real;
{------}
Procedure P1(k:Real);
Var i: Integer;
    s: Real;
Begin    {of P1}
  s:=0;
```

```
  For i:=1 to 20 do s:=s+k/i;
  y:=s;
End;     {of P1}
{------}
Procedure P2(k:Real; var r: real);
Var i: Integer;
    s: Real;
Begin    {of P2}
  s:=0;
  For i:=1 to 20 do s:=s+k/i;
  r:=s;
End;     {of P2}
{------}

Begin   {Of Program}
   x:=1;
   P1(x);
   P2(2*x, z);
   WriteLn('y=', y:4:2, '; z=', z:4:2)
End.   {Of Program}
```

В результате выполнения программы мы получим на экране следующее:

```
y=3.60; z=7.20
```

К о м м е н т а р и и к программе Lab7_02.

1. Из раздела описания процедур (P1 и P2) видно, что обе они передают в программу результаты своей работы через глобальные переменные. Только процедура P1 – обращаясь непосредственно к имени глобальной переменной (**y**:=s), а процедура P2, обращаясь к параметру-переменной (**r**:=s), передает данные фактическому параметру z (см. тело программы).

2. Первым фактическим параметром процедуры P2 является выражение `2*x`.

3. Рассмотренные способы передачи данных имеют свои недостатки.

Функция не может вернуть структурированных данных, например, массива.

Недостаток обмена через глобальные переменные – рост числа глобальных переменных. При этом, если мы обращаемся непосредственно к имени глобальной переменной (в процедуре P1), то во-первых, программу труднее понять, так как из заголовка процедуры не видно, в какой переменной подпрограмма оставляет результаты своей работы; а во-вторых, изменение имен глобальных переменных влечет необходимость изменения подпрограмм.

4. Необходимо помнить, что если имя локальной переменной совпадает с именем глобальной, то внутри соответствующей подпрограммы локальная переменная «закрывает» глобальную и из подпрограммы глобальной переменной «не видно». То есть, мы имеем две различные переменные с одним именем и при обращении к данному имени внутри подпрограммы, мы обращаемся к локальной переменной.

Пример 7. Модифицировать программу из примера 5.2, используя подпрограммы. Ввод параметров с клавиатуры оформить в виде процедуры, а подпрограмму вычисления членов ряда – в виде функции.

В ходе решения данной задачи были разработаны структурная схема алгоритма (рисунок 15) и программа `lab7_03`.

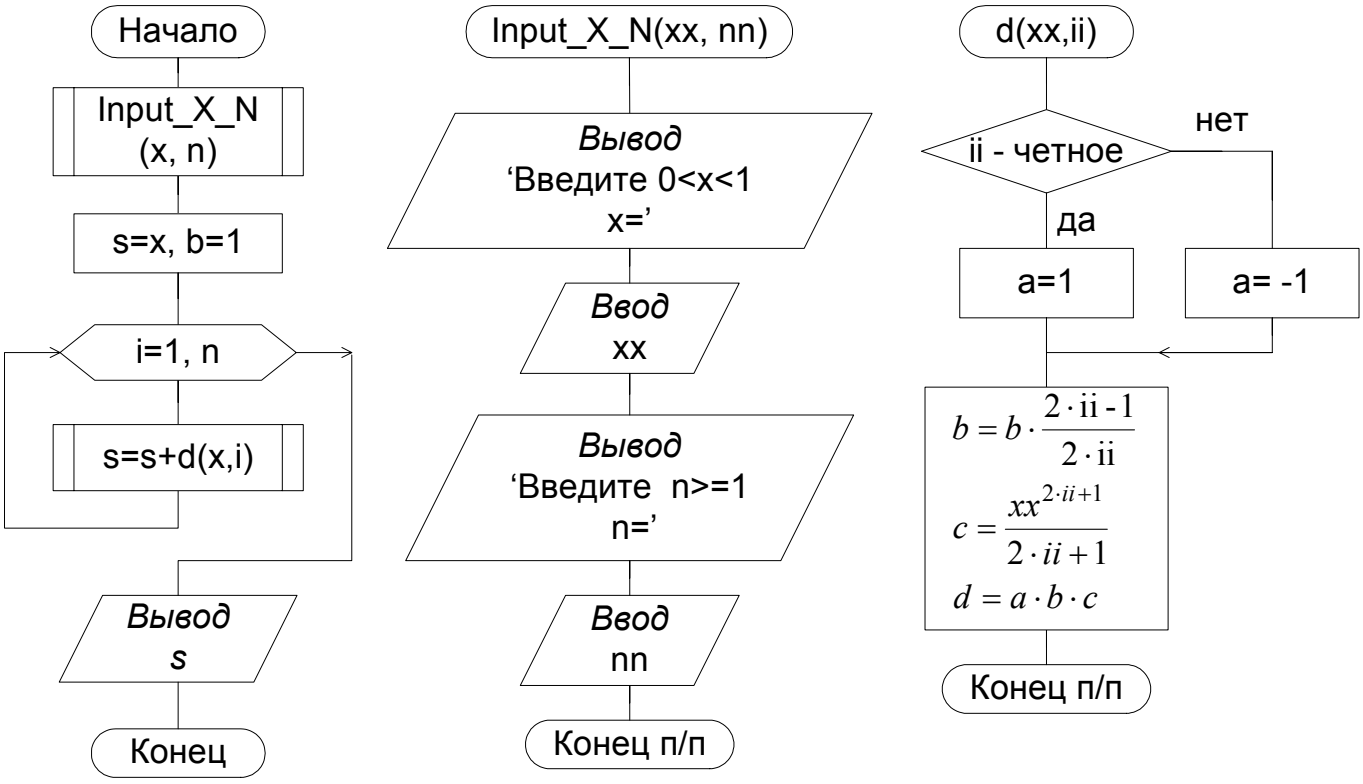


Рисунок 15

```
Program lab7_03;
Var b, x, s: Real;
    n,i: Integer;

Procedure Input_X_N(var xx: real; var nn: integer);
Begin     {Of Input_X_N}
   Write('Input 0<x<1.  x=');
   Read(xx);
   Write('Input n>=1.  n=');
   Read(nn);
End;      {Of Input_X_N}

Function d(xx:Real; ii:Integer):Real;
Var a, c: Real;
Begin     {Of d}
  If odd(ii) Then a:=-1
             Else a:=1;
  b:=b*(2*ii-1)/(2*ii);
```

```
  c:=exp((2*ii+1)*ln(xx))/(2*ii+1);
  d:=a*b*c;
End;      {Of d}

Begin     {Of Program}
   Input_X_N(x,n);
   s:=x;  b:=1;
   For i:=1 to n do s:=s+d(x,i);
   WriteLn('Summa=', s:8:5);
End.      {Of Program}
```

К о м м е н т а р и и к программе Lab7_03.

Процедура `Input_X_N(xx,nn)` служит для ввода данных с клавиатуры. Данные вводятся в глобальные переменные. Обращение к глобальным переменным производится через параметры-переменные. Имена формальных и фактических параметров сделаны различными для иллюстрации того, что они могут не совпадать.

Функция `d(x,i)` служит для вычисления слагаемых в сумме ряда. После вычисления очередного слагаемого информация из переменных `a`, `c` нигде не используется, поэтому указанные переменные сделаны локальными. Переменная `b`, напротив, должна учитывать все прошлые промежуточные вычисления. Поэтому, чтобы ее значение не терялось после завершения работы функции, она сделана глобальной. Обращение к этой переменной производится непосредственно через ее имя.

В отличие от нашей учебной программы, тексты других программ бывают объемными. В этих случаях, для улучшения читаемости, принято комментариями обозначать начало и конец подпрограмм и программы, как это показано выше.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Задание 1. Модифицировать программу из лабораторной работы № 3 с применение подпрограммы. Фрагмент программы, в котором вычисляется y(x) или z(t), оформить в виде функции.

Задание 2. Модифицировать программу из лабораторной работы № 6, оформив программу в виде трех процедур: ввода, обработки и вывода данных.

Лабораторная работа № 8

МОДУЛИ

ЦЕЛЬ РАБОТЫ: научиться создавать библиотечные модули.

Назначение и подключение модулей.

Модули являются инструментом для создания библиотек прикладных программ.

В Турбо Паскале имеются следующие стандартные библиотечные модули:

- System – основная библиотека;

- Dos – содержит ряд программ операционной системы и обработки файлов, которые не определены стандартом Паскаля;

- CRT – содержит подпрограммы управления текстовым выводом на экран дисплея, звуковым генератором, чтения клавиатуры;

- Graph – библиотека графических подпрограмм универсального назначения.

Подключение библиотек осуществляется следующим предложением, помещаемым сразу после заголовка программы:

```
USES <список модулей>;
```

где USES – зарезервированное слово (ИСПОЛЬЗУЕТ); <список модулей> состоит из имен подключаемых модулей, перечисляемых через запятую, например:

```
Uses CRT, Graph;
```

Модуль System подключается по умолчанию.

Создание модулей.

Для создания собственных модулей, необходимо знать *структуру модуля*:

```
UNIT <имя_модуля>;
     INTERFACE
<интерфейсная часть>
   IMPLEMENTATION
<исполнительная часть>
         BEGIN
<инициирующая часть>
         END.
```

Исходный текст модуля должен находиться в одноименном файле с расширением PAS и начинаться зарезервированным словом UNIT (модуль).

В *интерфейсной части*, открывающейся словом INTERFACE (интерфейс), содержатся объявления глобальных объектов модуля (типов, констант, переменных, подпрограмм), которые должны быть доступны основной программе или другим модулям. Объявления подпрограмм состоят только из их заголовков.

В *исполняемой части*, открывающейся словом IMPLEMENTATION (реализация), содержатся:

- описания подпрограмм, объявленных в интерфейсной части;

- объявления локальных для модуля объектов (типы, константы, переменные).

Описанию подпрограммы, объявленной в интерфейсной части, должен предшествовать заголовок подпрограммы, в котором можно опускать список формальных параметров и тип результата (для функции), так как они уже были объявлены в интерфейсной части.

Инициирующая часть модуля может отсутствовать вместе с начинающим ее словом BEGIN (остается слово END и следующая за ним точка – признак конца модуля). Данная часть обычно используется для подготовки к работе основной программы. Например: инициируются переменные, открываются файлы, устанавливается связь с другими ПК.

Пример 8. Создать модуль, содержащий три подпрограммы для нахождения: максимального, минимального и среднего арифметического значений элементов одномерного массива действительных чисел, а так же тестовую программу.

В ходе решения данной задачи были разработаны: структурные схемы алгоритмов тестирующей программы и подпрограмм (рисунок 16); исходный текст модуля `Arrays`, помещенный в файл `Arrays.pas`; программу `lab8` для тестирования модуля.

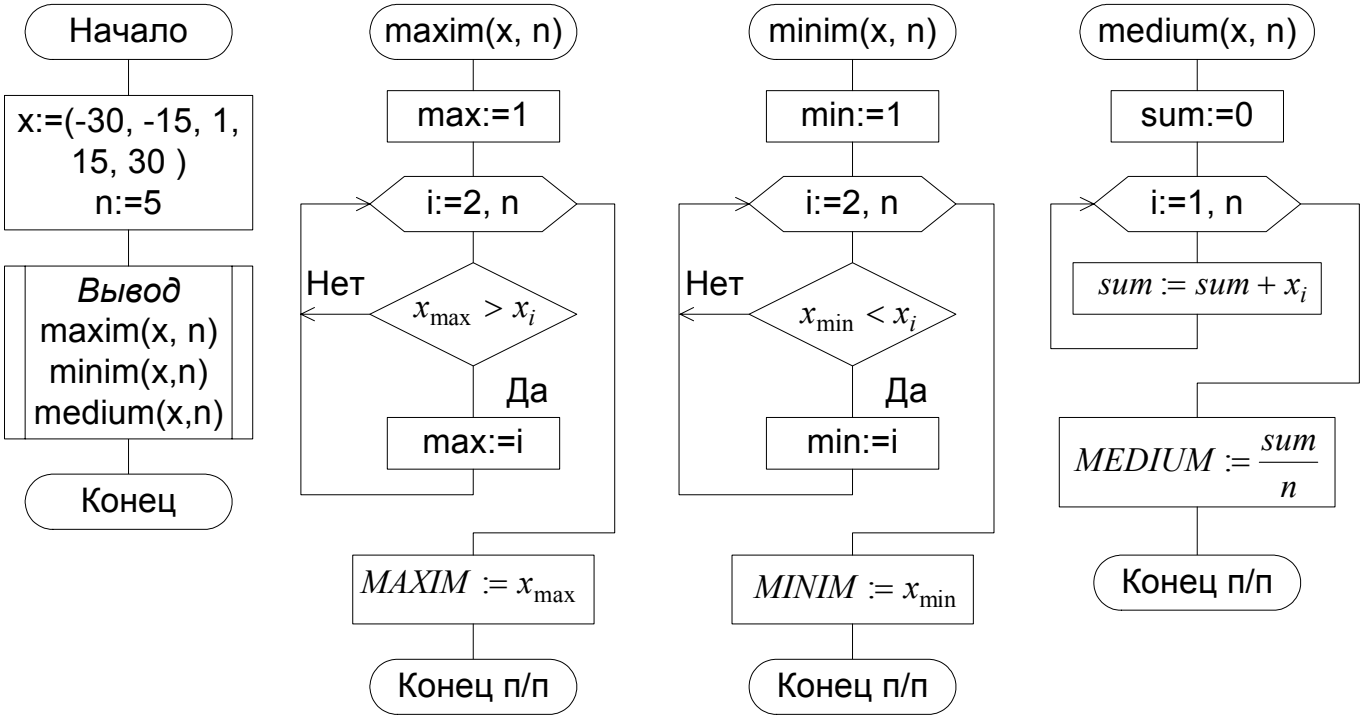


Рисунок 16

```
Unit Arrays;
{-----------------------------------------}
Interface
{-----------------------------------------}
Type
    ar = array[1..10] of real;
Function maxim(var x:ar; n:integer):real;
Function minim(var x:ar; n:integer):real;
Function medium(var x:ar; n:integer):real;
{-----------------------------------------}
Implementation
{-----------------------------------------}
FUNCTION Maxim;
var i,max:integer;
Begin
  max:=1;
  For i:=2 to n do If x[i]>x[max] Then max:=i;
  maxim:=x[max];
```

```
end;
FUNCTION Minim;
var i,min:integer;
Begin
  min:=1;
  For i:=2 to n do If x[i]<x[min] Then min:=i;
  minim:=x[min];
end;
FUNCTION Medium;
var i: integer;
    sum: real;
Begin
  sum:=0;
  For i:=1 to n do sum:=sum+x[i];
  medium:=sum/n;
end;
END.

Program lab8;
Uses Arrays, Crt;
Const x:ar=(-30, -15, 1, 15, 30 , 0, 0, 0, 0, 0);
      n=5;
var i: integer;
Begin
  ClrScr;
  WriteLn('Mimimum=', minim(x,n):5:2);
  WriteLn('Maximum=', maxim(x,n):5:2);
  WriteLn('Medimum=', medium(x,n):5:2);
end.
```

К о м м е н т а р и и.

Алгоритм. Подпрограммы (три) выполнены в виде функций. Тестирующая программа задает массив из пяти элементов. Значения, возвращаемые разработанными функциями, выводятся на экран. Функции имеют два параметра: массив и число элементов.

Текст модуля. В интерфейсном разделе объявлены:

- глобальный тип `ar` – одномерный массив из 10 действительных чисел;

- три функции, возвращающие действительные числа. Функции имеют два формальных параметра: параметр-переменная типа `ar` (куда передается исходный массив) и параметр-значение `n` – целое число, задающее количество обрабатываемых элементов массива. В нашем случае массив можно было передать и через параметр-значение, поскольку подпрограмме не нужно передавать обратно в программу обработанный массив. Однако, напомним, что если массив является фактическим параметром-значением, то в подпрограмму передается его *копия* и, когда необходимо повысить быстродействие программы (за счет исключение действий по копирова-

нию больших объемов информации), массив передают в подпрограмму в виде глобальной переменной через параметр-переменную.

В исполняемой части модуля описаны три функции: Maxim, возвращающая максимальное значение массива; Minim, возвращающая минимальное значение массива; Medium, возвращающая среднее арифметическое элементов массива.

Инициирующая часть модуля отсутствует.

Для улучшения читаемости заголовки разделов модуля выделены с помощью комментариев `{------}`.

Текст тестирующей программы Lab8.

Программа использует модули: Arrays (наш модуль), и Crt (стандартный. Из него мы используем процедуру очистки экрана ClrScr).

Входной массив задан глобальной типизированной константой. Типизированные константы объявляются в разделе описания констант следующим образом:

```
<имя_константы> : <тип_контанты> = <значение_константы>
```

Обратите внимание, что тип `ar` константы `x` определен в подключенном модуле `Arrays`. Константа `n` указывает на то, что из десяти элементов массива необходимо использовать только пять.

В теле программы после очистки экрана производится вывод на экран значений трех функций, описанных в модуле `Arrays`. Фактическими параметрами функций являются: глобальная константа-массив `x` (на месте параметра-переменной) и глобальная константа `n` (на месте параметра-значения).

Компиляция модулей.

В среде Турбо Паскаля существует три режима компиляции: *COMPILE*, *MAKE* и *BUILD*.

При компиляции программы в режиме *COMPILE* все нестандартные модули, упомянутые в предложении USES, должны быть предварительно откомпилировны (в том же режиме). В результате компиляции файла ИМЯ_МОДУЛЯ.PAS появится файл ИМЯ_МОДУЛЯ.TPU (от англ. Turbo Pascal Unit).

При компиляции программы в режиме *MAKE* компилятор проверяет наличие TPU-файлов для каждого объявленного модуля. Если какой-либо из файлов не обнаружен, система пытается отыскать одноименный файл с расширением PAS и приступает к его компиляции. Кроме того, в этом режиме система следит за возможными изменениями исходного текста любого используемого модуля. Если в PAS-файл (исходный текст модуля) внесены какие-либо изменения, то независимо от того, есть ли уже в каталоге соответствующий TPU-файл или нет, система осуществляет его компиляцию перед компиляцией основной программы.

В режиме *BUILD* система пытается найти и откомпилировать все исходные файлы модулей, объявленных в основной программе, независимо от наличия TPU-файлов.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Создать модуль, содержащий три подпрограммы и тестовую программу. Подпрограммами являются функции, вычисляющие значения по формулам, представ-

ленным в лабораторных работах № 1, № 2, и в задании № 2 из лабораторной работы № 5. Имя модуля должно начинаться с «UNIT» и заканчиваться номером варианта. Имена функций должны включать номер варианта и номер функции по порядку. Например, для варианта № 7 имя модуля и имена функций должны быть соответственно UNIT7, F7_1, F7_2, F7_3. В вариантах 22, 23 текст модуля будет содержать 4 функции, так как в задании № 2 из Л.Р.№5 требуется вычислить два значения.

Лабораторная работа № 9

ЗАПИСИ

ЦЕЛЬ РАБОТЫ: научиться разрабатывать программы с использованием записей и типизированных файлов.

Записи. *Запись*, – это структура данных, состоящая из нескольких компонентов, называемых полями записи. В отличие от массива, компоненты (поля) записи могут быть различного типа.

Тип «запись» объявляется следующим образом:

```
TYPE    <Имя типа> =    RECORD
                            <имя поля1>:<тип поля1>;
                            <имя поля2>:<тип поля2>;
                                    …
                            <имя поляN>:<тип поляN>
                        END;
```

Если тип нескольких полей совпадает, то имена полей можно перечислять через запятую.

Доступ к отдельному компоненту записи производится указанием составного имени: имени переменной и через точку имени поля.

Для того, чтобы не повторять часто одну и ту же общую часть составного имени, ее, указав один раз, можно далее умалчивать, используя оператор присоединения WITH:

WITH <общая_часть_составного_имени> DO <оператор>

Тогда в операторе, стоящем после DO, общая часть умалчивается и указываются только имена полей.

Типизированные файлы. В Л.Р. № 6 был рассмотрен пример работы со *стандартными текстовыми файлами*, которые определялись таким образом:

```
var <имя файловой переменной> = TEXT;
```

Кроме того, возможна работа с *типизированными файлами*, где единицей хранящихся данных являются данные любого типа Турбо Паскаля, кроме файлов. В этом случае файловые переменные определяются следующим образом:

```
var <имя файловой переменной> = FILE  OF <тип>;
```

В теле программы вывод данных в такой файл (или ввод из него) осуществляется через переменные соответствующего типа процедурами Read(), Write().

Пример 9.1 Разработать программу формирования списка студентов, включающего имя студента, дату рождения и распечатывания списка по месяцам рождения. Процедуры обработки файлов не использовать.

```
Program Lab9_1;
Type
   student = Record  {Запись}
               name: String[15];
               birthday : Record   {Вложенная запись}
                            day, month: Byte;
                            year: Word
                          End
             End;
var s: array[1..20] of student;
    i,j,n: Byte;
Begin
{-----Input the list---------------}
      Write('The number of students  =');
      ReadLn(n);
      for i:=1 to n do
         begin
            WriteLn('Student ', i);
            Write('Name=');
            ReadLn(s[i].name);
            WriteLn('Birthday:');
            Write('Year=');
            ReadLn(s[i].birthday.year);
            Write('Month (1-12) =');
            ReadLn(s[i].birthday.month);
            Write('Day=');
            ReadLn(s[i].birthday.day);
            WriteLn('* * *');
         end;

{------Output the list--------------}
       For j:=1 to 12 do
         for i:=1 to N do
            if s[i].birthday.month=j Then
                 WriteLn(s[i].name, '  ', s[i].birthday.day,
'.', s[i].birthday.month, '.', s[i].birthday.year);
End.
```

К о м м е н т а р и и к программе Lab9_1.

Тип student представляет собой запись, которая имеет два компонента:

- поле name (имя студента) – строка длиной 15 символов;

- поле birthday (день рождения) – запись, имеющая три поля: day (день), month (месяц), year (год).

Переменная s – массив из двадцати элементов типа student.

В теле программы сначала осуществляется формирование списка студентов: ввод числа студентов и данных о них. Затем, путем вложения двух счетных циклов, осуществляется вывод списка по месяцам.

Если нам потребуется проанализировать переменную (константу) S типа String, то к отдельному символу строки мы может обратиться по его *порядковому номеру N в строке* таким образом: S[N], а длину строки можно извлечь из нулевой позиции в строке: Ord(S[0]).

Пример 9.2. Разработать базу данных «Музей изобразительных искусств», включающую следующие сведения: инвентарный номер единицы хранения, вид экспоната, название, автор. Использовать процедуры обработки файлов.

Структурная схема алгоритма и комментарии к ней приведены в приложении 2.

```
Program lab9_2;
Uses Crt;
Type Art = Record
             Num: Word; {инвентарный номер}
             ArtExp: string[20]; {вид экспоната}
             Name: string[40];{название произведения}
             Autor: string[15]{автор произведения}
           end;
Const FName='Museun.dat';
var
    f: File of Art;
    Key: Char;

{Процедура создания файла}
Procedure Init;
var a: Art;
begin {of Init}
  With a do
    begin
       Num:=23105;
       ArtExp:='picture';
       Name:='The Lute Player';
       Autor:='Caravaggio';
    end;
  Rewrite(f);
  Write(f, a);
  Close(f);
```

```
end;  {of Init}

{процедура добавления записей}
Procedure Add;
var  FindRez: Boolean;{результат поиска дублирующих записей}
     a, b: Art;
begin {of Add}
  FindRez:=False;
  Reset(f);
  WriteLn('Adding of Content.');
  Write('Inventory Number='); ReadLn(b.Num);
  Write('Art of Exponat: '); ReadLn(b.ArtExp);
  Write('Name of Exponat: '); ReadLn(b.Name);
  Write('Autor: '); ReadLn(b.Autor);
  Repeat
     Read(f, a);
     If (a.Num=b.Num) and (a.ArtExp=b.ArtExp) and
(a.Name=b.Name) and (a.autor=b.autor) Then FindRez:= True;
  Until (FindRez=True) or Eof(f);
  If FindRez=False Then Write(f, b)
                   Else WriteLn('----Such Record already
exist----');
end;  {of Add}

{процедура вывода на экран содержимого базы данных}
Procedure Print;
var a: Art;
begin {of Print}
  Reset(f);
  WriteLn('Content of DataBase:');
  Repeat
     Read(f, a);
     WriteLn(a.Num,', ', a.ArtExp, ', ', a.Name, ', ',
a.autor)
  Until Eof(f);
end;  {of Print}

Begin {of program}
   Assign(f,FName);
{   Init;}
   ClrScr;
   Repeat
      WriteLn;
      WriteLn('Press key: "E" for EXIT, "A" - ADD Record,
"P" - PRINT DataBase');
```

```
      Key:=ReadKey;
      Case Key of
         'A', 'a': Add;
         'P', 'p': Print;
      end;
   Until (Key='E') or (Key='e');
   Close(f);
End.  {of program}
```

К о м м е н т а р и и к программе Lab9_2.

Функция **ReadKey** приостанавливает выполнение программы до нажатия любой клавиши и возвращает символ, соответствующий нажатой клавише.

Функция **EOF** (от англ. End Of File) возвращает значение ИСТИНА, если обнаружен конец файла.

Структура оператора выбора **CASE** такова:

```
CASE <ключ_выбора> OF <список_выбора> ELSE <оператор> END;
```

где <ключ_выбора> – выражение порядкового типа; <список_выбора> – одна или более конструкций вида: <константа_выбора>:<оператор>; <константа_выбора> – константа того же типа, что и <ключ_выбора>.

Оператор выбора работает следующим образом. Вначале вычисляется значение <ключ_выбора>, а затем в последовательности операторов <список_выбора> отыскивается такой, которому предшествует константа, равная вычисленному значению. Найденный оператор выполняется, после чего оператор выбора завершает работу. Если же в списке выбора не будет найдена константа, соответствующая вычисленному значению ключа выбора, управление передается оператору, стоящему за словом ELSE.

Часть ELSE <оператор> можно опускать. Тогда при отсутствии в списке выбора нужной константы никаких действий не производится.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Задание 1 (Программа 9_1). Не используя процедуры обработки файлов:

1. Разработать базу данных «Книжный магазин», включающую следующие сведения: название книги, автор, год издания, количество экземпляров, цена. Программа должна осуществлять поиск книги по названию.

2. Разработать базу данных продаваемых автомобилей, включающую следующие сведения: марка, модель, цвет, год выпуска, пробег, цена, телефон. Программа должна осуществлять поиск по марке или цене.

3. Разработать базу данных «Планеты Солнечной системы», включающую: название планеты, ее массу, диаметр, среднее расстояние от солнца, период обращения по орбите. Организовать поиск по названию планеты.

4. Разработать базу данных «Ювелирные изделия», включающую: наименование, цену, металл, камни, номер по каталогу. Осуществить поиск по любому полю.

5. Разработать базу данных магазина молочных продуктов, включающую: название товара, жирность, изготовителя, цену, дату изготовления. Осуществить поиск по любому полю.

6. Разработать базу данных «Магазин одежды: мужские костюмы», включающую: размеры (рост, обхват груди, обхват талии); цену; изготовителя; количество товара. Осуществить поиск по любому полю.

7. Разработать справочник «Химические элементы», включающую: название химического элемента, атомный номер, формулу, группу. Организовать поиск.

8. Разработать справочник «Химический состав и калорийность пищевых продуктов (на 1 кг. съедобной части)», содержащий следующие данные.

Название продукта	Содержание, %					Калорийность, кДж(ккал)/кг
	Белки	Жиры	Углеводы	Вода	Зола	

Организовать поиск.

9. Разработать базу данных «Абитуриенты», содержащую: фамилию имя, отчество абитуриента; его оценки по физике, математике, русскому языку. Вычислить проходной балл для заданного количества бюджетных мест.

10. Разработать справочник «Физические свойства металлов», включающий следующие данные: название металла, плотность, удельная теплоемкость при 20 °С, теплопроводность при 20 °С. Организовать поиск.

11. Разработать справочник «Электрические свойства металлов при 20 °С», включающий следующие данные: название металла; удельное электрическое сопротивление при 20 °С, Ом·мм^2/м; температурный коэффициент электрического сопротивления, 1/°С. Организовать поиск.

12. Разработать базу данных «Каталог музыкальных произведений», включающую: название произведения, автора, исполнителя, год записи, продолжительность звучания. Организовать поиск.

13. Разработать базу данных «Автобусные маршруты», содержащую: № маршрута, конечный пункт 1, конечный пункт 2, протяженность маршрута, время рейса. Организовать вывод списка маршрутов по конечному пункту.

14. Разработать телефонный справочник. По номеру телефона организовать поиск владельца и наоборот.

15. Дан список студентов и оценка каждого на экзамене. Подсчитать количество удовлетворительных оценок, хороших, отличных и средний балл в группе. Напечатать фамилии неуспевающих студентов.

16. Сформировать список студентов, в котором указать фамилию (имя, отчество); город, в котором получил среднее образование (номер школы, если обучался в Омске). Подсчитать сколько в группе иногородних студентов.

17. Сформировать запись «Зарезервированные слова TP»: слово и перевод. Подсчитать их количество. Организовать поиск по зарезервированному слову – перевод и наоборот.

18. Сформировать запись «английское слово – перевод». Вводя слово (английское или русское), найти перевод или выдать сообщение «нет в словаре». По возможности предусмотреть пополнение словаря.

19. Сформировать запись «Типы ТР»: имя, тип, операции, разрешенные в данном типе. Подсчитать количество разных операций и вывести для каждого типа и списки типов для каждой операции.

20. Сформировать пополняемую базу данных «Континент-страны», в которой указать столицы, численность населения, крупные города. Организовать поиск страны по городу, стран или городов на континенте.

21. Сформировать список граждан, в котором указать фамилию, имя, отчество, адрес, профессию. Организовать в программе выборку и подсчет граждан с одинаковой профессией.

22. Разработать базу данных «География. Реки», в которой указать название реки, местоположение, длину, глубину.

23. Разработать базу данных «Звезды», в которой указать название звезды, величину, созвездие.

24. Сформировать справочник «Диапазоны радиоволн», включающий: название диапазона, верхнюю и нижнюю границы диапазона длин волн, верхнюю и нижнюю границы полосы частот. Организовать поиск по названию диапазона или по длине волны.

25. Организовать справочник «Основные физические свойства важнейших полупроводниковых материалов», включающий химическую формулу, температуру плавления, теплопроводность, ширину запрещенной зоны. Организовать поиск по химической формуле.

26. Сформировать справочник «Магнитная восприимчивость парамагнитных материалов», включающий химическую формулу вещества и значение восприимчивости. Организовать вывод списка веществ, значение восприимчивости которых больше или меньше заданного значения.

27. Сформировать таблицу «Виды топлива», содержащую: название топлива; процентное содержание серы; процентное содержание золы; теплоту сгорания, МДж/кг; жаропроизводительность, °С. Организовать выборку веществ, теплота сгорания или жаропроизводительность которых больше заданных значений.

28. Сформировать таблицу «Основные параметры пьезоэлектрических материалов», включающую: название вещества, плотность, скорость звука, диэлектрическую проницаемость, пьезомодуль. Организовать вывод списка веществ по заданному диапазону значений какого-либо из параметров.

29. Разработать базу данных «Загрязнители воздуха и воды», включающую: 1) название вещества; 2) его предельно допустимую концентрацию (ПДК) в атмосферном воздухе населенных мест; 3) его ПДК в воде водных объектов хозяйственно-питьевого и культурно-бытового водопользования. Организовать поиск по названию вещества.

30. Разработать базу данных «Календарь игр российской футбольной премьер-лиги». Организовать выборку по дате или по участнику.

Задание 2 (Программа 9_2).

Дополнить программу 9_1 заполнением и обработкой файлов.

Лабораторная работа № 10

ГРАФИЧЕСКИЕ СРЕДСТВА

ЦЕЛЬ РАБОТЫ: научиться пользоваться некоторыми средствами, представленными в модуле Graph.

Работа с шаблонами.

Для составления программ с использованием графических средств удобно пользоваться шаблонами, предлагаемыми справочной службой Турбо Паскаля.

Пример.

1. Выбрав в меню Help→Index, попадаем на индексную страницу.
2. Набрав на клавиатуре `lin`, видим, что подсветка остановилась на слове “Line”.
3. Нажав ENTER, попадаем на страницу с описанием процедуры Line.
4. На данной странице может быть представлен пример программы с использованием процедуры Line. Для доступа к примеру программы в другом варианте Help необходимо перейти по гиперссылке **Line.PAS**, находящейся в конце страницы.
5. С помощью мыши выделяем текст программы.
6. Запоминаем выделенный фрагмент в буфер, выбирая меню Edit→Copy или комбинацией Ctrl+Ins.
7. Клавишей Esc закрываем окно справочной службы.
8. Выбрав в меню File→New, создаем новый файл и сохраняем его.
9. Через меню Edit→Paste или комбинацией Shift+Ins вставляем выделенный фрагмент:

```
uses Crt, Graph;
var Gd, Gm: Integer;
begin
  Gd := Detect;
  InitGraph(Gd, Gm, '');
  if GraphResult <> grOk then
    Halt(1);
  Randomize;
  repeat
    Line(Random(200), Random(200), Random(200), Ran-
dom(200));
  until KeyPressed;
  Readln;
  CloseGraph;
end.
```

10. В параметрах процедуры `InitGraph` между апострофами вставьте путь до директории с BGI-файлами (драйверами графических адаптеров). Например:

```
InitGraph(Gd, Gm, 'D:\pascal\tp\bgi');
```

Так для работы с монитором VGA в данной директории должен находиться файл EGAVGA.BGI.

11. Проверим, что в установках имеется ссылка на директорию с файлом GRAPH.TPU. Для этого в меню Options→Diredtories… в строке *Unit directories* для нашего компьютера должен содержаться путь D:\pascal\tp\units (для вашего компьютера путь, скорее всего, – другой). По окончании редактирования параметров в данном окне нажимаем кнопку OK.
12. Если на компьютере установлена операционная системаWindows, то для Работы с графикой переходим в полноэкранный режим. (Вход и выход из данного режима осуществляется комбинацией клавиш Alt+Enter).
13. Осуществляем запуск программы на выполнение. После этого должна появиться квадратная область, заполняемая линиями. Выход из программы – нажатием ENTER.
14. Убедившись таким образом, что данный шаблон работает и все установки сделаны правильно, убираем из программы все лишнее:

```
uses Graph;
var Gd, Gm: Integer;
begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'D:\pascal\tp\bgi');
  if GraphResult <> grOk then
    Halt(1);

{Здесь должен располагаться основной текст нашей будущей
программы}

  Readln;
  CloseGraph;
end.
```

Этот текст можно сохранить как рабочий шаблон. Дадим к нему необходимые комментарии.

Процедура `InitGraph` автоматически распознает графический адаптер, установленный на ПК, загружает и инициализирует соответственный графический драйвер и *переводит адаптер в графический режим*. Графический и текстовый режимы не совместимы. Поэтому, при переходе в графический режим с экрана пропадает все, что было выведено в текстовом режиме. Закрытие графического режима и возвращение в текстовый осуществляется процедурой `CloseGraph`. При этом выведенная на экран графическая информация так же пропадает. И чтобы этого не произошло в следующее мгновение после вывода графического изображения, перед процедурой `CloseGraph` стоит процедура `Readln` (без аргументов – ждет нажатия клавиши Enter).

Если при работе в графическом режиме на экран требуется вывести текст, то это необходимо производить не с помощью процедур Write и WriteLn (они работают в текстовом режиме), а с помощью процедур OutText или OutTextXY из модуля Graph.

Описание процедур, функций и констант.

В графическом режиме можно адресоваться к любому элементу (пикселю) растрового графического экрана. Координатами пикселей являются два целых положительных числа: в горизонтальном направлении (ось X) и в вертикальном (ось Y). Координата X увеличивается слева направо, координата Y – сверху вниз. Координаты левого верхнего угла экрана равны 0,0; координаты правого нижнего угла можно получить с помощью следующих двух функций:

`GetMaxX`: Integer – возвращает *максимальную горизонтальную координату* графического экрана;

`GetMaxY`: Integer – возвращает *максимальную вертикальную координату* графического экрана.

Кроме указанных выше процедур и функций в лабораторной работе используются следующие процедуры.

`PutPixel``(X,Y: Integer; Color: Word)` – выводит *точку* цветом Color с координатами X, Y. Цвет можно задавать следующими стандартными константами, описанными в модуле Graph, (указывая имя или число):

```
Const
```

Black	= 0; {Черный}	DarkGray	= 8; {Темно-серый}
Blue	= 1; {Синий}	LightBlue	= 9; {Светло-синий}
Green	= 2; {Зеленый}	LightGreen	=10;{Светло-зеленый}
Cyan	= 3; {Голубой}	LightCyan	=11;{Светло-лубой}
Red	= 4; {Красный}	LightRed	=12;{Розовый}
Magenta	= 5; {Фиолетовый}	LightMagenta	=13;{Малиновый}
Brown	= 6; {Коричневый}	Yellow	=14;{Желтый}
LightGray	= 7; {Светлосерый}	White	=15;{Белый}

Например:

```
PutPixel(56, 100, 15);
PutPixel(10, 100, LightRed);
```

`Line``(X1, Y1, X2, Y2: Integer)` – рисует *линию* от точки X1, Y1 до точки X2, Y2. Например:

```
Line(0,0, 200, 300);
```

Рисование линии производится *основным цветом*, который устанавливается следующей процедурой.

`SetColor``(Color: Word)` – устанавливает основной цвет, которым будет производиться рисование.

Пример:

```
SetColor(Blue);
```

Bar(X1, Y1, X2, Y2: Integer) – рисует *полосу* (заштрихованную прямоугольную область), используя текущий цвет и стиль. Точка X1, Y1 и точка X2, Y2 задают *диагональ* полосы, рисунок 17. Пример:

```
Bar(300, 300, 20, 20);
```

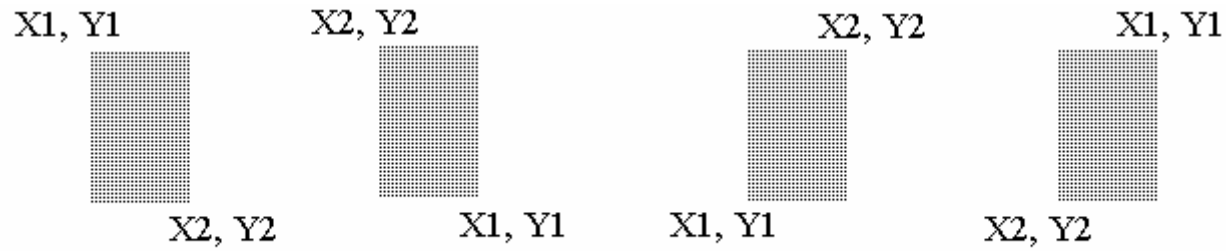


Рисунок 17

SetFillStyle(Pattern, Color: Word) – устанавливает образец штриховки (Pattern) и ее цвет (Color).

В следующей таблице приведены константы, задающие тип штриховки и определенные в модуле Graph:

Таблица 7

Константа	Значение	Тип штриховки
EmptyFill	0	Нет штриховки
SolidFill	1	Сплошная штриховка
LineFill	2	--- штриховка
LtSlashFill	3	/// штриховка
SlashFill	4	/// штриховка толстыми линиями
BkSlashFill	5	\\\ штриховка толстыми линиями
LtBkSlashFill	6	\\\ штриховка
HatchFill	7	Штриховка вклетку
XHatchFill	8	××× штриховка
InterleaveFill	9	××× штриховка частая
WideDotFill	10	Штриховка редкими точками
CloseDotFill	11	Штриховка частыми точками
UserFill	12	Штриховка задается пользователем

Примеры:

```
SetFillStyle(LineFill, Red);
SetFillStyle(3, 4);
```

Bar3D(X1,Y1,X2,Y2:Integer; Depth:Word; Top:Boolean) – рисует *параллелепипед*, у которого заштрихована передняя грань. В процедуре BAR3D к параметрам, имеющимся в процедуре BAR, добавляются: Depth (толщина боковой грани, в пикселях) и Top (наличие верхней грани). Параметр Top может задаваться стандартными константами: TopOn (рисовать верхнюю грань) и TopOff (не рисовать верхнюю грань). Тип и цвет штриховки передней грани устанавливается процедурой

SetFillStyle (как и для процедуры BAR). Цвет линий, которыми вычерчиваются ребра параллелепипеда, устанавливается процедурой SetColor. Пример:

```
Bar(300, 300, 20, 20, 10, TopOn);
```

Ellipse(X,Y:Integer; StAngle,EndAngle,XR,YR:Word)– вычерчивает *дугу эллипса* с центром эллипса X, Y, начиная от стартового угла StAngle и кончая конечным углом EndAngle (против часовой стрелки, рисунок 18). Радиусы эллипса: RX (по оси X), RY (по оси Y). Рисование ведется основным цветом.

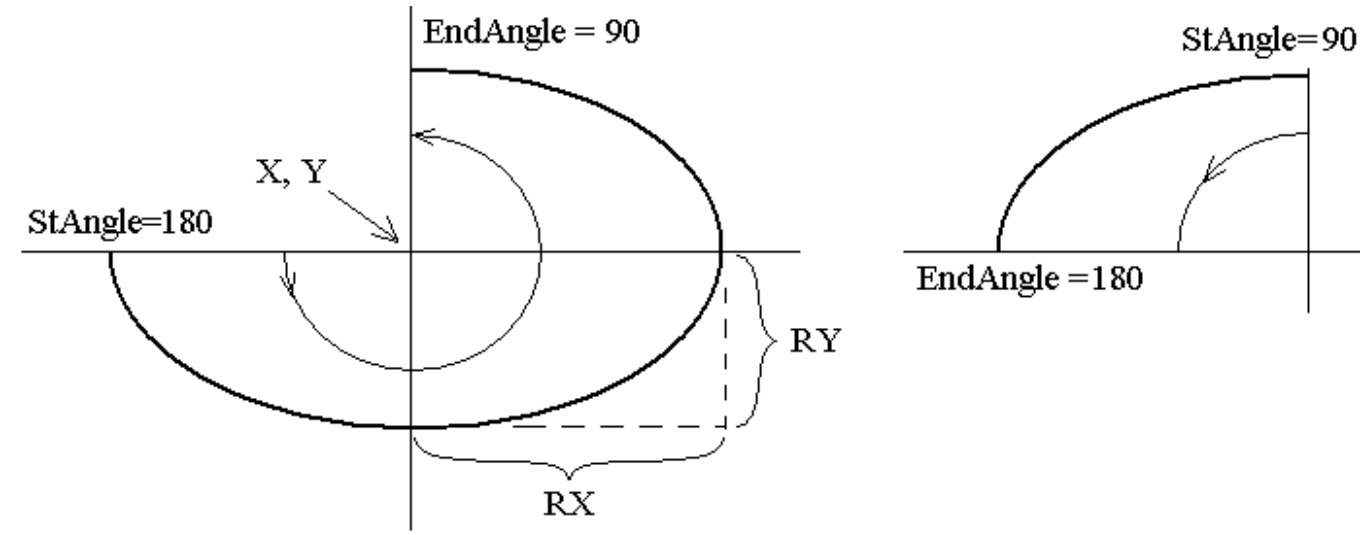


Рисунок 18

FloodFill(X,Y:Integer; Border:Word)– штрихует замкнутую область, содержащую внутреннюю точку с координатами X, Y, и ограниченную линией с цветом Border. Например, рисунок 19а:

```
FloodFill(150, 233, Red);
```

Рисунок 19

Типичными ошибками в применении данного оператора являются следующие:

- штрихуемая область не замкнута, рисунок 19б;
- ошибочно указана внутренняя точка, рисунок 19в;
- цвет линий, ограничивающих область, не соответствует цвету, указанному в процедуре FloodFill;

- закрашивание замкнутой области начинается до того, как она нарисована (процедура FloodFill ставится в тексте программы раньше, чем последняя процедура, замыкающая область.

Если трудно найти причины, по которым закрашивание осуществляется неправильно, то для выявления причины, изображенной на рисунке 19в, на время отладки программы вместо процедуры FloodFill можно поставить процедуру PutPixel с теми же координатами.

Пример 10. Вычертить, пользуясь указанными выше средствами, имя “САНЯ”.

Для решения задачи берем тетрадный лист, наносим на него координаты, отсчитывая узлы решетки (количество клеток), и рисуем требуемый текст, рисунок 20а. Координаты концов линий рекомендуется выбирать с дискретностью 1 или 0,5 клетки.

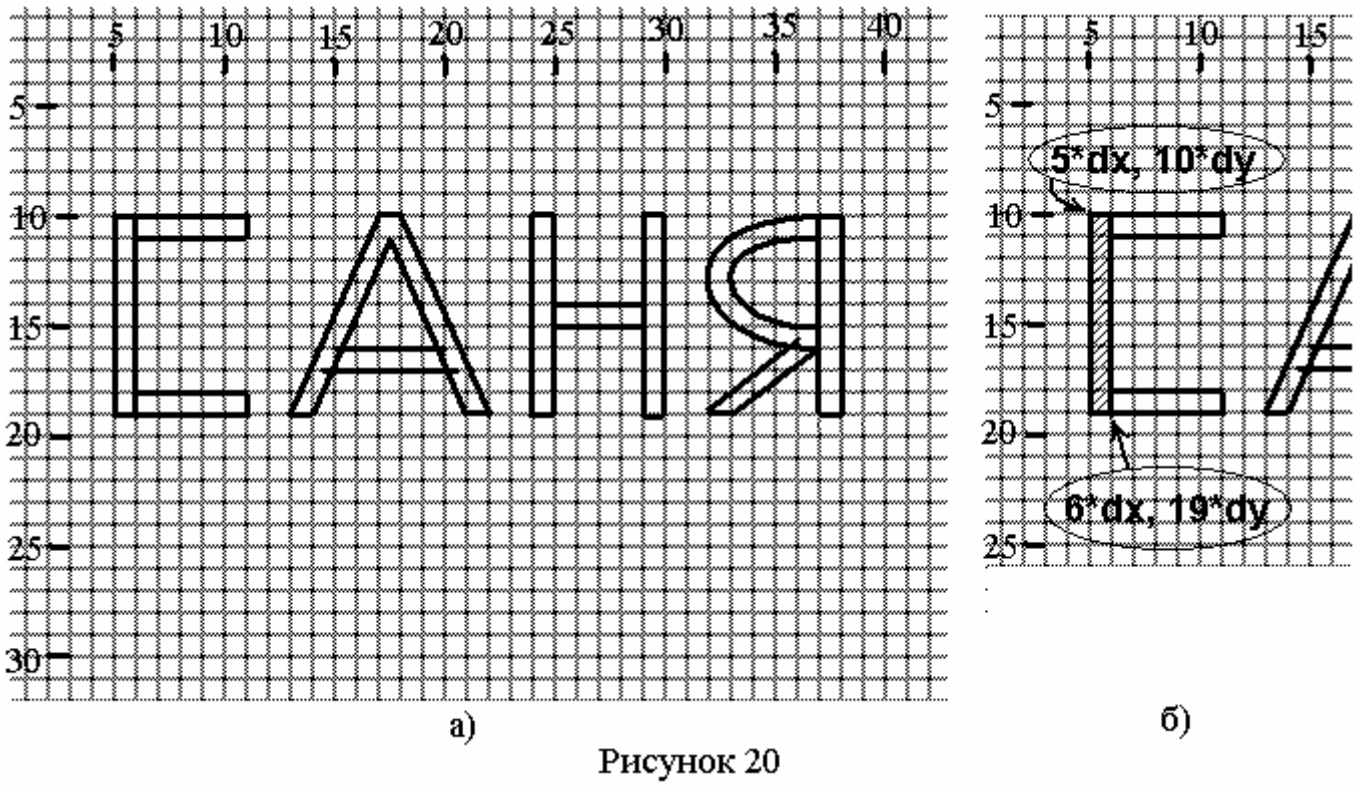


Рисунок 20

Положим, что максимальные координаты, отмеченные на листе бумаги, соответствуют координатам, возвращаемым функциями GetMaxX и GetMaxY. Тогда количество пикселей, приходящихся на одну клетку, равно:

dx=GetMaxX div 40 (по координате X);

dy=GetMaxY div 30 (по координате Y).

Количество пикселей, приходящихся на половину клетки, равно соответственно:

dx2=dx div 2 (по координате X);

dy2=dy div 2 (по координате Y).

При этом не составит труда быстро отсчитать по клеткам координаты первой (вертикальной) полосы, рисунок 20б:

```
Bar(5*dx,10*dy,6*dx,19*dy);
```

Продолжая таким же образом и далее, получим следующую программу.

```
Program Lab10;
uses Graph;
var Gd, Gm: Integer;
    dx, dy, dy2, dx2: Integer;
    Color: Byte;
begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'D:\pascal\tp\bgi');
  if GraphResult <> grOk then
    Halt(1);

  dx:= GetMaxX div 40;
  dy:= GetMaxY div 30;
  dy2:=dy div 2;
  dx2:=dx div 2;

  Color:=LightBlue;
  SetColor(Color);
  SetFillStyle(4,Color);

{S}
   Bar(5*dx,10*dy,6*dx,19*dy);
   Bar(6*dx,10*dy,11*dx,11*dy);
   Bar(6*dx,18*dy,11*dx,19*dy);

{A}
   Line(13*dx,19*dy,14*dx,19*dy);

   Line(21*dx,19*dy,22*dx,19*dy);
   Line(17*dx,10*dy,18*dx,10*dy);
   Line(13*dx,19*dy,17*dx,10*dy);
   Line(22*dx,19*dy,18*dx,10*dy);
   Line(14*dx,19*dy,17*dx+dx2,11*dy);
   Line(21*dx,19*dy,17*dx+dx2,11*dy);
   Line(15*dx,16*dy,20*dx,16*dy);
   Line(14*dx+dx2,17*dy,20*dx+dx2,17*dy);
   FloodFill(17*dx+dx2,10*dy+dy2,Color);
   FloodFill(17*dx,16*dy+dy2,Color);

{N}
   Bar3d(24*dx,10*dy,25*dx,19*dy,dx2,TopOn);
   Bar3d(29*dx,10*dy,30*dx,19*dy,dx2,TopOn);
   Bar3d(25*dx,14*dy,29*dx,15*dy,dx2,TopOn);
```

```
{YA}

   Bar(37*dx,10*dy,38*dx,19*dy);
   Line(33*dx,19*dy,32*dx,19*dy);
   Line(33*dx,19*dy,37*dx,16*dy);
   Line(32*dx,19*dy,36*dx,15*dy+dy2);
   Ellipse(37*dx, 13*dy, 90,270, 4*dx, 2*dy);
   Ellipse(37*dx, 13*dy, 90,270, 5*dx, 3*dy);

   FloodFill(34*dx,11*dy,Color);
   FloodFill(35*dx,17*dy,Color);

  Readln;
  CloseGraph;
end.
```

ЛАБОРАТОРНОЕ ЗАДАНИЕ

Пользуясь, по возможности, процедурами: Bar, Bar3D, Line, Ellipse, FloodFill, написать программу рисования на экране Вашей фамилии таким же образом, как показано в примере 10.

Библиографический список

Основная литература.

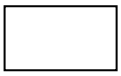
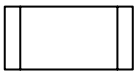
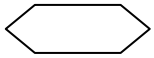
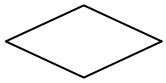
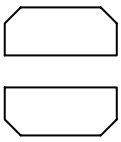
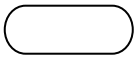
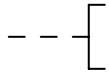
1. Фаронов В.В. Турбо Паскаль (в 3-х книгах). Книга 1. Основы Турбо Паскаля. – М.: Учебно-инженерный центр «МВТУ-ФЕСТО ДИДАКТИК», 1992 – 304 с.
2. Шафеева О.П. Основы программирования. Турбо Паскаль 7.0: Конспект лекций. – Омск, 2000.

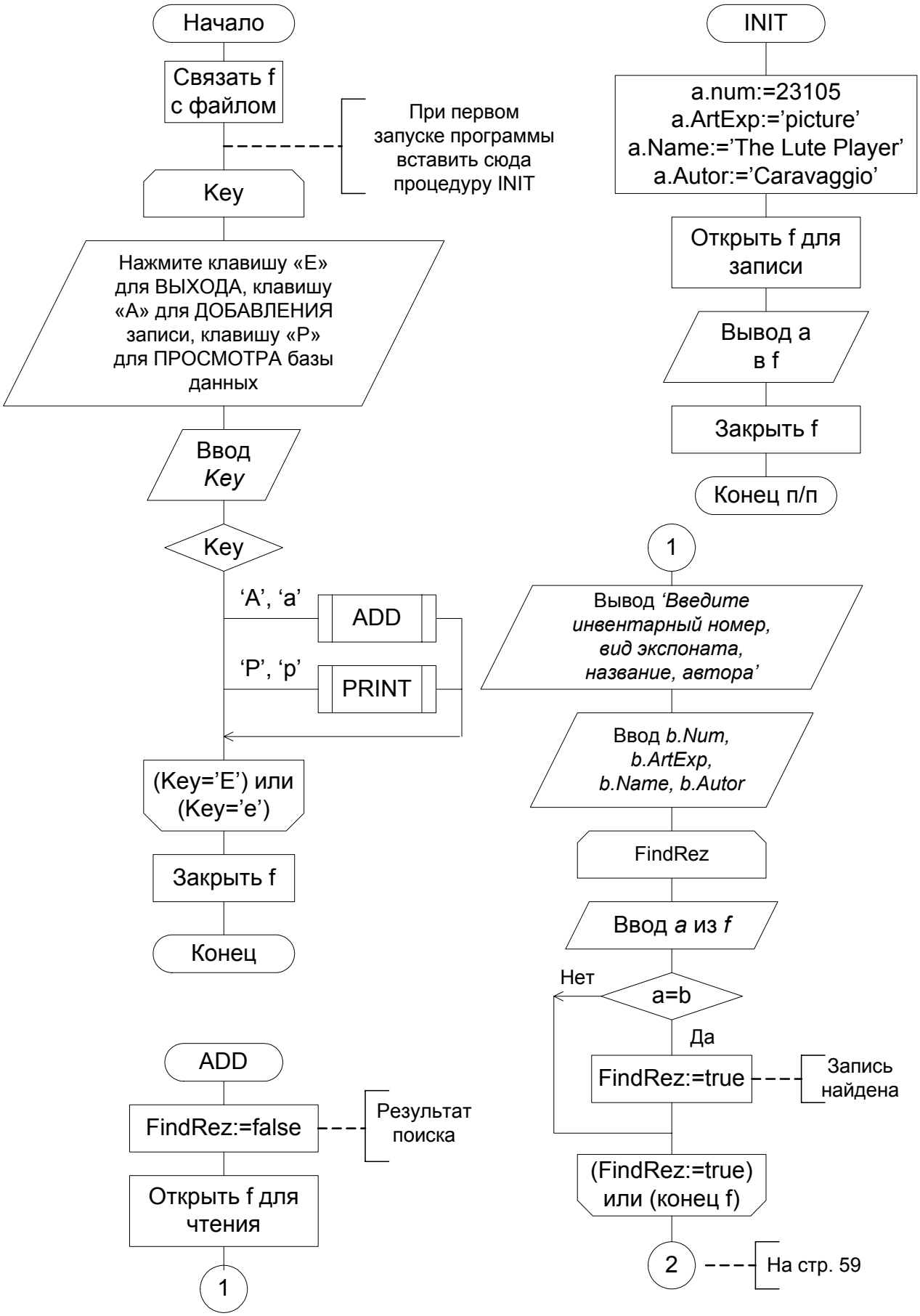
Дополнительная литература.

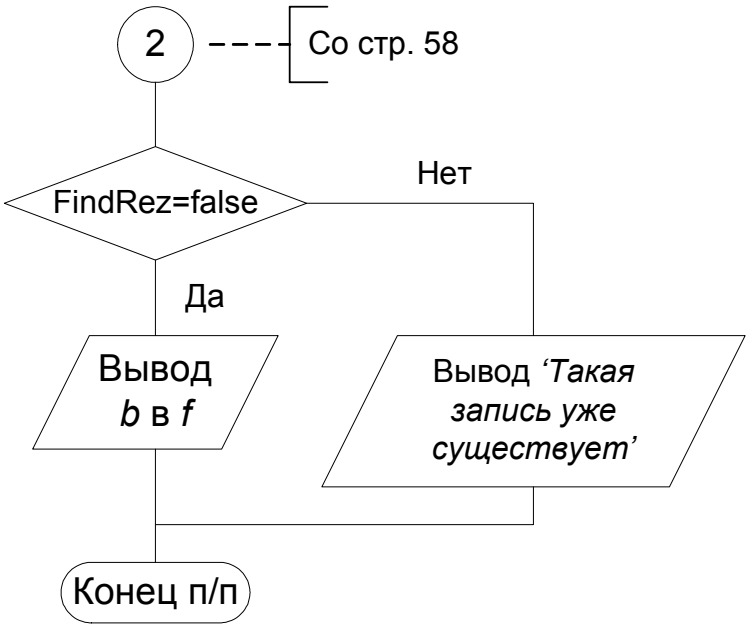
3. Иванова Г. С. Технология программирования.: Учебник для вузов. – М.: изд-во МГТУ им. Н. Э. Баумана, 2002. – 320 с.
4. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб.: Питер, 2002. – 464 с.

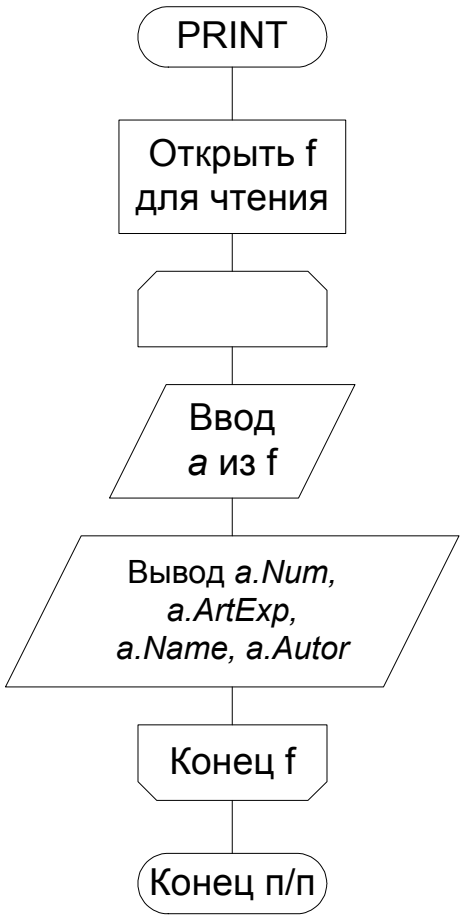
Приложение 1

Обозначения условные графические в структурных схемах алгоритмов (ГОСТ 19.701-90)

Символ	Наименование	Назначение
	Данные	Определяет ввод или вывод на внешнее устройство или любой носитель данных.
	Процесс	Отражает обработку данных: выполнение определенной операции или группы операций.
	Предопределенный процесс	Отображает предопределенный процесс, состоящий из одной или нескольких операций программы, которые определены в другом месте (подпрограмме, модуле).
	Подготовка цикла	Отражает инициализацию и модификацию параметра для управления циклом со счетчиком.
	Решение	Описывает проверку условия и выполняет переключение по одному из условий. Имеет один вход и два или более альтернативных выходов, один из которых активизируется после вычисления условия внутри символа.
	Граница цикла	Состоит из двух частей: начала и конца цикла. Обе части имеют один и тот же идентификатор. Изменение значения идентификатора, условия для выполнения или завершения помещаются внутри символов в начале или в конце цикла.
	Соединитель	Используется для обрыва линии и продолжения ее в другом месте. Должен содержать уникальное обозначение.
	Терминатор	Определяет начало и конец структурной схемы алгоритма программы или подпрограммы.
	Комментарий	Используется для добавления пояснительных записей. Связывается с символом или группой символов, обведенных пунктиром.
	Основная линия	Отображает последовательность выполнения действий в алгоритме.


Начало
Связать f с файлом
При первом запуске программы вставить сюда процедуру INIT
Key
Нажмите клавишу «E» для ВЫХОДА, клавишу «A» для ДОБАВЛЕНИЯ записи, клавишу «P» для ПРОСМОТРА базы данных
Ввод Key
Key
'A', 'a'
ADD
'P', 'p'
PRINT
(Key='E') или (Key='e')
Закрыть f
Конец
ADD
FindRez:=false
Результат поиска
Открыть f для чтения
1
INIT
a.num:=23105
a.ArtExp:='picture'
a.Name:='The Lute Player'
a.Autor:='Caravaggio'
Открыть f для записи
Вывод a в f
Закрыть f
Конец п/п
1
Вывод 'Введите инвентарный номер, вид экспоната, название, автора'
Ввод b.Num, b.ArtExp, b.Name, b.Autor
FindRez
Ввод a из f
Нет
a=b
Да
FindRez:=true
Запись найдена
(FindRez:=true) или (конец f)
2
На стр. 59





К о м м е н т а р и и к алгоритму.

Если файл базы данных museum.dat еще не был создан, то в тело основной программы необходимо включить процедуру INIT, которая создаст файл и внесет в него первую запись. При последующих сеансах работы с программой процедура INIT должна быть исключена, иначе будут уничтожены все данные, введенные позднее.

Элемент «решение» в теле основной программы образует алгоритмическую конструкцию переключателя с тремя и более альтернативными выходами. В данном случае конструкция имеет три выхода. Первый выход активизируется, если переменная Key равна ’a’ или ‘A’ (в латинском алфавите); второй выход – если Key=’p’ или ‘P’; в остальных случаях (по третьему выходу) никаких действий не производится. В языке Паскаль данной алгоритмической конструкции соответствует оператор выбора CASE.

Процедура ADD добавляет новые записи в базу. При этом, введенная с клавиатуры запись сравнивается со всеми имеющимися в базе и в случае совпадения файл не дополняется.

Процедура PRINT выводит содержимое базы данных на дисплей.