С.В. АРТЕМОВА, Т.И. ЧЕРНЫШОВА,
Н.Г. ЧЕРНЫШОВ

ИНФОРМАТИКА

ИЗДАТЕЛЬСТВО ТГТУ

Министерство образования и науки Российской Федерации

ГОУ ВПО "Тамбовский государственный технический университет"

С.В. АРТЕМОВА, Т.И. ЧЕРНЫШОВА, Н.Г. ЧЕРНЫШОВ

ИНФОРМАТИКА

Рекомендовано УМО по образованию в области радиотехники, электроники, биомедицинской техники и автоматизации в качестве учебно-методического пособия для студентов специальности 210200 дневного и заочного отделений

Издание второе, стереотипное

Тамбов
Издательство ТГТУ
2006

УДК 519.682
ББК З81я73-1
А86

Рецензент
Доктор технических наук, профессор ТВВАИУ
Д.А. Дмитриев

Артемова, С.В.
А86 Информатика : учебно-методическое пособие. 2-е изд., стер. / С.В. Артемова, Т.И. Чернышова, Н.Г. Чернышов. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2006. – 40 с. – 100 экз. – ISBN 5-8265-0474-9.

Изложены основные понятия и методы информатики. Содержатся задания и практические примеры реализации программ с использованием наиболее распространенных языков СИ и Паскаль.

Предназначено для выполнения лабораторных и контрольных работ по курсу "Информатика" для студентов специальности 210200 "Проектирование и технология радиоэлектронных средств".

УДК 519.682
ББК 32.81я73-1

ISBN 5-8265-0474-9

Учебное издание

АРТЕМОВА Светлана Валерьевна,
ЧЕРНЫШОВА Татьяна Ивановна,
ЧЕРНЫШОВ Николай Генрихович

ИНФОРМАТИКА

Учебно-методическое пособие

Издание второе, стереотипное

Редактор Т.М. Глинкина

Компьютерное макетирование Е.В. Кораблевой

Подписано в печать 22.06.2006
Формат 60 × 84/16. Бумага офсетная. Гарнитура Times New Roman.
2,3 уч.-изд. л. Тираж 100 экз. Заказ № 344

Издательско-полиграфический центр ТГТУ
392000, Тамбов, Советская, 106, к. 14

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Учебно-методическое пособие «Информатика» предназначено для выполнения лабораторных и контрольных работ студентами всех форм обучения. В нем излагаются основные понятия и методы информатики. Главное внимание уделяется программированию. Изложение основ программирования проводится с использованием наиболее распространенных языков СИ и Паскаль.

Большинство программ создаются для решения какой-нибудь задачи. Решение задачи достигается обработкой информации или данных. Поэтому для написания любой программы необходимо знать, как:

- ввести информацию в программу, т.е. осуществить ввод;
- сохранять информацию или данные;
- задать правильные команды обработки данных;
- получить данные из программы, т.е. осуществить вывод.

Написать и упорядочить свои команды можно так, чтобы:

- некоторые из них выполнялись при выполнении некоторого условия или ряда условий – условное выполнение;
- другие выполнялись некоторое число раз – циклы;
- повторяющиеся собирались в отдельные части, которые могут быть выполнены в нескольких местах программы – подпрограммы.

Здесь перечислены семь основных понятий программирования: ввод, данные, операторы, вывод, условное выполнение, циклы и подпрограммы. Этот список неполный, но он содержит основные понятия, присущие всем программам.

Все языки программирования, включая СИ и Паскаль, имеют свои особенности, но когда необходимо быстро изучить новый язык, можно посмотреть, как он реализует эти семь элементов, и начать работать.

Материал в пособии излагается в последовательности от простого к сложному. Все тексты приведенных в пособии программ являются переносимыми, их можно использовать с любым компилятором, способным реализовать на компьютере консольное приложение.

Пособие построено таким образом, что студент довольно быстро привлекается к самостоятельному составлению осмысленных программ, что позволяет использовать его не только для дневной и заочной, но и для дистанционной формы образования.

Изложение сопровождается большим числом примеров. Помимо этого оно снабжено задачами для самостоятельного решения по вариантам.

Лабораторная работа 1

СОСТАВЛЕНИЕ ПРОСТЕЙШИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОГРАММ ЛИНЕЙНОЙ СТРУКТУРЫ, ПРОГРАММИРОВАНИЕ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ, ОПЕРАЦИИ ВВОДА – ВЫВОДА

Цель работы: изучить программирование алгоритмов линейной структуры и алгоритмических выражений, использование операторов ввода-вывода, присвоения и математических функций.

ЗАДАНИЕ

1 Составить блок-схему алгоритма линейной структуры.

2 Изучить особенности программирования арифметических выражений на алгоритмических языках СИ и Паскаль.

3 Составить программу для заданного алгоритма. При составлении программы следует пользоваться операторами ввода-вывода, позволяющими выводить на экран приглашения-подсказки ввода информации.

4 Ввести текст программы в компьютер, используя редактор.

5 Провести расчет по программе.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Алгоритм линейной структуры (следование). Блочные символы в этой структуре располагаются на схеме в том же порядке, в каком должны быть выполнены предписываемые ими действия. Такой порядок исполнения действий называется естественным [2].

Выражение представляет собой компактную запись, указывающую, какие операции надо провести над данными, чтобы получить требуемое значение. Различают арифметические, логические и строковые выражения.

Арифметическое выражение образуется из операндов и операций [2].

Операндами арифметического выражения являются константы, переменные, элементы массивов, обращения к функциям, заключенные в скобках выражения. Над этими операндами с помощью знаков арифметических операций могут быть произведены следующие действия:

– сложение +;
– вычитание –;
– умножение *;
– деление /;
– остаток от деления % в СИ и mod в Паскале;
– инкремент (увеличение на 1)++ – в СИ и inc в Паскале;
– декремент (уменьшение на 1) -- – в СИ и dec в Паскале.

Выражения, заключенные в скобки, вычисляются в первую очередь. Операции одного порядка выполняются последовательно слева направо, за исключением операций возведения в степень. Несколько последовательных операций возведения в степень выполняются справа налево.

При записи арифметических операций рекомендуется обращать внимание на следующие моменты:

– При записи произведения констант и переменных нельзя опускать знак умножения. Запись ab воспринимается ЭВМ как имя переменной ab, а не произведение переменных a и b. Произведение a и b записывается как a*b.

– Не допускается запись двух следующих друг за другом знаков арифметических операций. Например, запись выражения a/–b неверна. Надо записать a/(–b).

– Аргумент арифметических функций всегда заключается в скобки (запись sin x неверна, правильно – sin(x)).

– Промежуточные результаты вычисления арифметических выражений не должны выходить из допустимой области значений.

– При записи вещественных констант целую и дробную часть отделяет десятичная точка, а не запятая (2.132 – правильно, 72,15 – неправильно).

– При написании сложных арифметических выражений необходимо проконтролировать равенство числа открывающих и закрывающих скобок.

– В Паскале нет операции возведения в степень, при возведении в квадрат следует использовать умножение (x*x), а в остальных случаях пользоваться формулой $x^y = \exp(y * \ln(x))$, для вычисления которой в Паскале есть стандартные функции exp и ln.

ОПЕРАТОРЫ ВВОДА-ВЫВОДА В СИ И ПАСКАЛЕ

Программа, использующая перечисленные ниже функции ввода-вывода должна включать в себя файл stdio.h с помощью команды препроцессора

```
#include <stdio.h>
```

Файл stdio.h содержит:

1 Определение типа данных FILE.

2 Определение параметров, используемых в макровызовах и вызовах библиотечных функций.

Форматизированный ввод и вывод осуществляют два семейства функций: scanf обеспечивает форматизированный ввод, а семейство printf – форматизированный вывод.

fscanf – форматизированный ввод из потока,
scanf – форматизированный ввод из потока stdin,
sscanf – форматизированный ввод из строки СИ,

int scanf (const char *format [....]);
fprintf – форматизированный вывод в поток,
printf – форматизированный вывод в поток stdout,
sprintf – форматизированный вывод в строку СИ.
int printf (const char *format [,argument,....]);

Символ формата	Тип выводимого объекта
%c	char (символ)
%s	строка
%d, %i	int
%f	float/double

Например:

```
#include <stdio.h>
  main()
   {
    int name;
    printf("введите целое число");
    scanf("%d",&name);
    printf("Вы ввели число %d ",name);
  }
```

Программа, написанная на Паскале, должна иметь следующую структуру:

заголовок программы;
раздел описания меток;
раздел описания констант;
раздел описания типов;
раздел описания переменных;
раздел описания процедур и функций;

```
Begin
 оператор 1;
 оператор 2;
 .........
 оператор N
End.
```

ОПЕРАТОРЫ ВВОДА-ВЫВОДА В ПАСКАЛЕ

Оператор ввода Read имеет вид:

```
Read(СП);,
```

где СП – список переменных, подлежащих вводу. Оператор Read производит ввод данных с клавиатуры через стандартный файл Input.

При наборе данных при вводе их следует отдалить друг от друга одним пробелом. При выполнении оператора Read конец строки, на которой размещаются входные данные, приравнивается к разделяющему данные пробелу.

Например, операторы

```
Read (a); Read (b); Read (c);
```

работают также как оператор

```
Read(a,b,c);
```

где переменные a – целого, b – вещественного, c – символьного типа.

Если требуется, чтобы a=2,b=0.125,c="*", то данные надо набирать таким образом:

2_0.125_* <возвр.>

Оператор ReadLn в отличие от оператора Read после своего выполнения автоматически осуществляет переход к следующей строке экрана.

Например, для ввода данных в соответствии с операторами

```
ReadLn(a); ReadLn(b); ReadLn(c);
```

Операторы вывода Write и WriteLn имеют вид:

```
Write (СП) ;
WriteLn (СП) ;
```

где СП – список вывода, который может содержать, помимо имен переменных, подлежащих выводу, символьные константы. Кроме того, при вводе данных можно указывать количество позиций, отводимых под вводимое значение. Для переменных целого или символьного типа после имени переменной ставится двоеточие и количество позиций, отводимых под ее значение.

Например, оператор

```
Write("_a=",a:3,"_c=",c:4);
```

выведет на экран

a=__2_c=___*

Для переменных вещественного типа можно указать общее число позиций для дробной части. При этом значение переменной выводится в виде с фиксированной точкой.

Например

Write(b: 5: 3); выведет 0.125;

Write(b: 5: 2); выведет 0.13

Если для вещественной переменной не указывается число позиций вообще, то ее значение выводится в экспоненциальной форме (с плавающей точкой).

Операторы Write,WriteLn производят вывод на экран дисплея через стандартный файл Output в том случае, если часть выходных данных требуется вывести не на экран, а на печатающее устройство, потребуется ввести в программу следующие изменения:

1) В заголовок программы, кроме имен файлов Input, Output включить имя файла (например Print или любое другое имя), через который будет производится вывод на печать.

2) В раздел описания переменных Var включить описание:

Print:Text;(файл текстового типа)

3) Во всех операторах вывода на печать указать имя файла, через который будет идти вывод на печать:

Write(Print,СП);

4) Перед первым оператором Write открыть файл Print оператором

Rewrite(Print,"LP:");

Если одни и те же данные необходимо вывести на печать и на экран, то в программе должны остаться оба оператора:

Write(СП);

Write(Print,СП).

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 Получить вариант задания в соответствии с порядковым номером в журнале преподавателя.

1) $y=\dfrac{\sqrt{1+e^{\sqrt{x}}+\cos x^2}}{\left|1-\sin^3 x\right|}+\ln\left|2x\right|;$

2) $y=\dfrac{0{,}5x^2-1}{\sqrt{x}}+\dfrac{\left|10-e^{0{,}5x^2+1}\right|}{\ln 2x-1};$

3) $y=\dfrac{\sqrt{2+3\cos^2 2x+2}-1}{2^{1+\ln x}-\left|1-\sin\sqrt{2+x}\right|};$

4) $y=\dfrac{\cos\sqrt{x+1}}{2\left|1-\sqrt{x}\right|+3\dfrac{1+e^x}{2^{x+1}-1}};$

5) $y=\dfrac{x+\sin 0{,}5x+\sqrt{\ln x}}{2+3\cos x}+\dfrac{e^{2x-0{,}5}}{2+x};$

6) $y=\dfrac{\sqrt{x^3}-\ln x}{\cos 2x^2+\left|x-3\right|}+\dfrac{3(x-2)^2}{\ln x+2};$

7) $y=\dfrac{2\ln x\cos 2x-3\dfrac{(x+1)^2}{x-1}}{2+\sqrt{x}};$

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

9) $y=\dfrac{\sin 3x+e^{1=x^2}}{2^{x+1}x+1-e^{5x^2-1}};$

10) $y=\dfrac{\left|1+\cos\sqrt{x}-\ln^2 2x\right|}{\sqrt{3x+4}-2^{0{,}5x-1}};$

11) $y=\dfrac{\sqrt{3+\ln x+15-x}}{1+\sin\dfrac{2+x^2}{1+x}};$

12) $y=\dfrac{e^{x-\sqrt{x}}}{\ln\dfrac{x+5}{x+1}-2\dfrac{x^3+x}{1+\left|\sin x\right|}};$

13) $y=\dfrac{2\ln\sqrt{x}-3\dfrac{x+1}{2x+3}}{\sin^2 2x+\left|e^{2x}-3x^2\right|};$

14) $y=\dfrac{\sqrt{2+0{,}5x^2+e^{x+2}}}{x^3-3(2x-1)^2}+\sqrt{x};$

15) $y=\dfrac{5+3\ln\sqrt{1+2x}}{2e^{1+0{,}5x^2}-\left|1-2x\right|+3};$

16) $y=\dfrac{x^2-3\sqrt{x+1}+\ln 2x^2}{3+\sin\dfrac{2x^2+1}{2+\sqrt{x}}}.$

2 Составить программы вычисления значений арифметических выражений на языках СИ или Паскаль. Обозначить в текстах программ порядок выполнения арифметических операций.

3 Ввести и отладить программу.

4 Выполнить расчеты по программе.

ПРИМЕРЫ ПРОГРАММ

Программа Паскаль

```
{Составление простейших вычислительных программ линейной структуры}
{Программирование арифметических выражений.}
{Операции ввода-вывода.}
uses crt; {Подключается библиотека CRT}
var x,y:Real;
begin
  ClrScr;{Производится очистка экрана}
  WriteLn('Выполнил Поляков Д.Г., КТФ, гр. Р-11');
  WriteLn('Введите аргумент вычисляемого выражения');
  ReadLn(x); { Считывается вводимое число }
  y := ( Sin(3*x) + Exp(1+x*x) ) / ( Exp((x+1)*Ln(2)) + 1 – Exp(5*x*x–1) );
  WriteLn('Выводится результат вычисляемого выражения');
  WriteLn(y:4:5); { Форматный вывод }
  ReadKey; { Задержка экрана }
end.
```

Программа на СИ

```
// Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами
#include<math.h> // Библиотека математических функций

void main(void) // Главная функция (основной блок программы)
{
 // Раздел объявления локальных переменных
   double f; // Функция
   float x; // Аргумент
 // Раздел объявления констант
   const float e=2.7183; // Число e-основание натурального логарифма
   clrscr(); // Очистка экрана
   printf("Составитель:  Грибков Алексей Николаевич");
   printf("\n\t\tФакультет КТФ, группа Р-11");
   printf("\n\nВведите x: %f",&x);
 // Вычисление значения функции
   f=((sqrt(1+pow(e,sqrt(a)+cos(a*a))))/
     (fabs(1-sin(a)*sin(a)*sin(a))))+(log(fabs(2*a)));
   printf("Значение функции: %lg",f); // Вывод на экран значения функции
   getch(); // Задержка экрана до нажатия любой клавиши
}
```

СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

1 Цель лабораторной работы.
2 Блок-схема алгоритма программы.
3 Текст программы.
4 Результат расчета.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Что такое идентификатор? Каковы сходства и различия идентификаторов в языках СИ и Паскаль?
2 Каким арифметическим выражениям соответствуют следующие записи на языке Паскаль?
y=a+d/c**2 z=sin(x)**4/2
f=(cos(x))**2–exp(q**2–1) s=(z+2)/(3+e**5)-sin((x)**2).
3 В каких единицах выражаются аргументы тригонометрических функций?
4 Найдите ошибки в следующих строках программы.
x=0.25*(cos(2z–3)+a/sin(f), y=2.14*a+exp x-sin(x-20),
f=2p+q/–v+g**n+cos(((x))).

Литература [2, 3].

Лабораторная работа 2

СОСТАВЛЕНИЕ ПРОСТЕЙШИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОГРАММ РАЗВЕТВЛЕННОЙ СТРУКТУРЫ

Цель работы: Изучить программирование алгоритмов разветвленной структуры.

ЗАДАНИЕ

1 Составить блок-схему алгоритма разветвленной структуры.

2 Изучить особенности программирования алгоритмов разветвленной структуры на алгоритмических языках СИ или Паскаль.

3 Запрограммировать полученный алгоритм.

4 Ввести текст программы в компьютер.

5 Выполнить расчеты по программе.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Алгоритмы разветвленной структуры (ветвление). Это такая схема, в которой предусмотрено разветвление указанной последовательности действий на два направления в зависимости от итога проверки заданного условия [1].

Логические операции

СИ	Паскаль

Логические функции

НЕ !	.not.
И &&	.and.
ИЛИ !	.or.

Логические отношения

Равно =	.eq.
Не равно !=	.ne.
Больше >	.gt.
Меньше <	.lt.
Больше или равно >=	.gt.
Меньше или равно <=	.le.

ОПЕРАТОРЫ ВЕТВЛЕНИЯ В СИ

К операторам ветвления в СИ относятся if, if else, ?.
switch и goto. Общий вид операторов ветвления следующий :

```
if ( логическое выражение )
  оператор;
```

```
if (логическое выражение)
  оператор_1
else
  оператор_2
```

```
<логическое выражение> ? <выражение_1>: <выражение_2>;
```

Если значение логического выражения истинно ,то вычисляется выражение_1, а в противном случае вычисляется выражение_2.

```
switch(выражение целого типа )
{
 case значение_1:
   последовательность_операторов_1;
   break;
case значение_2:
   последовательность_операторов_2;
   break;
   ...
case значение _n:
   break;
default:
   последовательность _операторов n+1
}
```

Ветку default можно не описывать. Ветка выполняется, если одно из вышестоящих условий не удовлетворительно.

Используя вложенные операторы if else, можно построить логический эквивалент конструкций if then elsif else.

Например:

```
if (a>b)
  printf(''\n больше'');
```

```
else if (a==b)
  printf (''\nA и B равны'');
else if (a>c && d<e)


 printf(''\n Условие 3 удовлетворено'');
else
   printf(''\n Действие по умолчанию'');
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 Получить вариант задания

1) $f(z)=\begin{cases}1, & z>10;\\ 2z-1, & 0\le z\le 10;\\ \left|1-3z\right|, & z<0;\end{cases}$

2) $f(z)=\begin{cases}2z, & z\le 0;\\ 1+\sqrt{z}, & 0<z<5;\\ -2z+1, & z\ge 5;\end{cases}$

3) $f(z)=\begin{cases}-1, & z<0;\\ 2z, & 0\le z<5;\\ z; & z\ge 5;\end{cases}$

4) $f(z)=\begin{cases}2+z, & z<2;\\ z^2, & 2\le z\le 4;\\ 1+0{,}5z, & z>4;\end{cases}$

5) $f(z)=\begin{cases}4, & z\ge 7;\\ z, & -2<z<7;\\ \left|z+2\right|, & z\le -2;\end{cases}$

6) $f(z)=\begin{cases}z^2, & z>5;\\ 0{,}5z, & 2<z\le 5;\\ -1, & z\le 2;\end{cases}$

7) $f(z)=\begin{cases}0{,}5z, & z>3;\\ 2, & 0\le z\le 3;\\ 1/z, & z<0;\end{cases}$

8) $f(z)=\begin{cases}\sqrt{z}, & z>0;\\ z, & -3<z\le 0;\\ -5, & z\le -3;\end{cases}$

9) $f(z)=\begin{cases}z^2, & z>2;\\ 0{,}5z, & -1<z\le 2;\\ \left|z+1\right|, & z\le -1;\end{cases}$

10) $f(z)=\begin{cases}10, & z\ge 10;\\ 2z+1, & 0<z<10;\\ 0, & z\le 10;\end{cases}$

11) $f(z)=\begin{cases}1/z, & z\ge 3;\\ 3z, & 1\le z<3;\\ z^2, & z<1;\end{cases}$

12) $f(z)=\begin{cases}z-2, & z\ge 3;\\ 0{,}5z, & -2<z<3;\\ -2, & z\le -2;\end{cases}$

13) $f(z)=\begin{cases}0{,}5z^2, & z<0;\\ 2z+1, & 0\le z<4;\\ z-2, & z\ge 40;\end{cases}$

14) $f(z)=\begin{cases}\sqrt{z}, & z>5;\\ 2, & -1\le z\le 5;\\ z+1, & z<-1;\end{cases}$

15) $f(z)=\begin{cases}5, & z>4;\\ 3z, & 0\le z\le 4;\\ \left|z\right|, & z<0;\end{cases}$

16) $f(z)=\begin{cases}\ln z, & z>2;\\ 3z-2, & -3<z\le 2;\\ 2+\left|z\right|, & z\le -3.\end{cases}$

2 Составить программы вычисления значений арифметических выражений на языках СИ и Паскаль.
3 Обозначить в текстах программ порядок выполнения арифметических операций.
4 Выполнить расчет по программе.

СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

1 Цель лабораторной работы.
2 Блок-схема алгоритма программы.
3 Текст программы.
4 Результат расчета.

Пример программы на Паскале для расчета значения функции, заданной интервально

```
{ Составление простейших вычислительных программ }
{ Разветвленной структуры }
{ Программирование арифметических выражений. }
{ Операции ввода-вывода. }

uses CRT; { Подключается библиотека CRT }
var x, y: Real;
```

```
  s: Char;  { Переменная символьного типа }
  f: Boolean; { Переменная логического типа }
begin
  ClrScr; { Производится очистка экрана }
  WriteLn('Составил Поляков Д.Г., КТФ, гр. Р-11 ');
  WriteLn; { Вывод пустой строки }
  repeat { Цикл с постусловием }
    f := False; { Флаговая переменная }
    WriteLn('Введите аргумент вычисляемого выражения');
     ReadLn(x); { Считывается вводимое число }
     if (x>10) then y:=1
     else if (x>=0) and (x<=10) then y:=2*x–1
     else if (x<0) then y:=Abs(1–3*x);
     WriteLn('Выводится результат вычисляемого выражения');
     WriteLn(y:4:5); { Форматный вывод }
      WriteLn('Вычислять ли функцию для другого аргумента? Y/N');
       s:=ReadKey; { Считывается символ с клавиатуры }
       WriteLn;
       if s in ['y','Y'] then f := True;
  until f = False;
end.
```

Пример программы на СИ: Программа вычисляет значение функции y=f(z), в зависимости от значения переменной z:

```
                // Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами
#include<math.h> // Библиотека математических функций

void main(void) // Главная функция (основной блок программы)
{
 // Раздел объявления локальных переменных
   int z, p; // z-аргумент, p-флаговая переменная
   char key; // Символьная переменная
 // Раздел объявления констант
    const int true=1; // Истина
    const int false=0; // Ложь
    clrscr(); // Очистка экрана
    printf("Составитель:  Грибков Алексей Николаевич");
    printf("\n\t\tФакультет КТФ, группа Р-11");
   while(p!=true) // Цикл продолжается до тех пор, пока значение переменной
  {         // p не станет равным true (истина)
      printf("\n\nВведите z: ");
      scanf("%i", &z); // Ввод значения переменной z
   // Вычисление значения y=f(z)
    if (z>10) printf("y=1, z>10");
    if (z>=0 && z<=10) printf("y=%i, 0=<z<=10",2*z–1);
    if (z<0) printf("y=%i, z<0",abs(1-3*z));
   // Запрос на повторение расчетов с другими параметрами
    printf("\n\nХотите произвести расчеты еще раз: y/n");
    key=getch(); // Считывание символа с клавиатуры в переменную key
    if (key!='y' && key!='Y') p=true;
    else p=false;
  }
}
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Как выполняется оператор if?
2 Какое выражение называется логическим?
3 Как обозначаются логические операции в СИ и в Паскале?
4 Каких основных правил следует придерживаться при программировании разветвляющихся алгоритмов?

Литература [2, 3].

Лабораторная работа 3

СОСТАВЛЕНИ ПРОСТЕЙШИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОГРАММ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Цель работы: получение навыков программирования итерационных алгоритмов и алгоритмов циклической структуры.

ЗАДАНИЕ

1 Составить блок-схему алгоритма циклической структуры.

2 Изучение особенности программирования алгоритмов циклической структуры на алгоритмических языках СИ и Паскаль.

3 Запрограммировать полученный алгоритм.

4 Ввести текст программы в компьютер.

5 Выполнить расчет по программе.

Используя оператор цикла:

а) найти с заданной точностью сумму элементов убывающей последовательности, т.е. вычислять сумму до тех пор, пока n-й элемент последовательности не станет меньше заданной точности. Точность вводится с клавиатуры. Вывести на экран сумму элементов, значение n-го элемента и количество элементов, необходимых для подсчета суммы с заданной точностью;

б) найти наименьшее значение m, для которого произведение не будет превышать некоторую заданную величину. Эту величину ввести с клавиатуры. Вывести на экран значение m и полученное произведение.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

При решении ряда задач возникает необходимость многократного повторения однотипных действий при различных значениях параметров, определяющих эти действия. Алгоритмы, реализующие такие вычисления, называются циклическими, а повторяющиеся участки вычислений – циклами. Использование циклов позволяет выполнить большие объемы вычислений при помощи компактных программ. Различают циклы с заданным и неизвестным числом повторений. К последним относятся итерационные циклы с требуемой точностью. Бывают циклы с проверкой условия перед началом выполнения тела цикла, по окончании выполнения тела или внутри тела цикла. Ниже будут рассмотрены все указанные типы циклов для СИ [3].

ПРОГРАММИРОВАНИЕ ЦИКЛОВ В СИ

Цикл while

Синтаксис цикла while (пока) таков:

```
while (условие _выражение)
    оператор
```

В цикле типа while проверка условия производится перед выполнением тела цикла. Если результат вычисления условного выражения не равен нулю, то выполняется оператор (или группа операторов). Перед входом в цикл while в первый раз обычно инициализируют одну или несколько переменных для того, чтобы условное выражение имело какое-либо значение. Оператор или группа операторов, составляющих тело цикла, должны, как правило, изменять значения одной или нескольких переменных, входящих в условное выражение, с тем чтобы, в конце концов, выражение обратилось в нуль и цикл завершился. Пример:

```
printf("\n Отвечает yes no (y/n) :  ")
scanf("%c", &ch) ;
while(ch!='y' && ch!= 'n')
 {
    printf( "\n Отвечает yes no (y/n) :  ")
    scanf("%c", &ch) ;
 }
```

Цикл do while

В цикле do while проверка условия осуществляется после выполнения тела цикла. Синтаксис цикла:

```
do
оператор
while (условное_выражение);
```

Цикл do while прекращает выполняться, когда условное выражение обращается в нуль (остается ложным). Как и для цикла while, для цикла do while можно описать ситуации, приводящие к выходу из цикла:

*Условное выражение обратилось в 0

*Внутри цикла встретился break.

*Внутри цикла выполнен оператор return.

Пример:

```
mm=10;
do
index++;
while (index<mm);
```

Цикл for

Наиболее общей формой цикла в СИ является цикл for.

Цикл for – это общая и более мощная форма, чем аналогичный цикл в языке Паскаль.

Конструкция for выглядит следующим образом:

```
for([ необязательное выражение1];
  [ необязательное выражение2];
  [ необязательное выражение3])
оператор
```

Каждое из трех выражений можно опускать. Хотя в принципе каждое из этих выражений может быть использовано программистом как угодно, обычно первое выражение служит для инициализации индекса, второе для проверки на окончание цикла, а третье выражение – для изменения значения индекса.

Формально это правило можно описать так:

1 Если первое выражение присутствует, то оно вычисляется.
2 Вычисляется второе выражение (если оно присутствует).
3 Исполняется тело цикла.
4 Вычисляется третье выражение (если оно присутствует).
5 Переход к п. 2.

Появление в любом месте цикла оператора continue приводит к немедленному переходу к шагу 4.

Пример:

```
int i , mas[4];
for( i=0; i < 4;i++)
  scanf(''%d'', &mas[i];
```

ПРОГРАММИРОВАНИЕ ЦИКЛОВ В ПАСКАЛЕ

```
  while условное выражение do
    оператор;
```

for с шагом 1 (инкрементный)

```
for индекс:=выражение1 to выражение2 do оператор;
```

for с шагом –1 (декриментный)

```
for индекс:=выражение1 downto выражение2 do оператор;
```

```
repeat
  операторы
   until ложное условное выражение.
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 Получить вариант задания:

1) $\sum_{n=1}^{\infty}\frac{n+1}{2n!}$;
2) $\sum_{n=1}^{\infty}\frac{1}{2n+n!}$;
3) $\sum_{n=5}^{\infty}\frac{\sqrt{n+2}}{1+\sqrt{(n+1)!}}$;
4) $\sum_{n=1}^{\infty}\frac{5+n}{3n!-1}$;
5) $\sum_{n=5}^{\infty}\frac{\sqrt{n}}{(n-3)!}$;
6) $\sum_{n=0}^{\infty}\frac{2n+1}{(3n+1)!}$;
7) $\sum_{n=1}^{\infty}\frac{2}{\sqrt{n!+4}}$;
8) $\sum_{n=2}^{\infty}\frac{n-1}{2n!-2}$;
9) $\prod_{n=1}^{m}\frac{\sqrt{n!}+8n}{3n-2}$;
10) $\prod_{n=1}^{m}0{,}5(2+0{,}1!)$;
11) $\prod_{n=0}^{m}\frac{(2n+1)!}{\sqrt{n+2}}$;
12) $\prod_{n=2}^{m}\frac{0{,}5n!}{0.1n+1}$;
13) $\prod_{n=0}^{m}\frac{(n+2)!}{n+1}$;
14) $\prod_{n=0}^{m}\sqrt{(3n+1)!}$;
15) $\prod_{n=1}^{m}\frac{3n!-1}{n+1}$;
16) $\prod_{n=1}^{m}\frac{\sqrt{n!+10n}}{3n-2}$.

2 Составить программы вычисления значений суммы и произведений ряда на языках СИ и Паскаль.
3 Ввести и отладить программу.
4 Выполнить расчеты по программе.

Пример программы на Паскале

{Сумма членов ряда с заданной точностью $\sum_{n=1}^{\infty}\frac{n+1}{2n!}$;}

```
uses crt; { Подключается библиотека ввода-вывода }
var k, eps : Real;
  n, i, x : Integer;
begin
  ClrScr; { Производится очистка экрана }
  WriteLn('Выполнил Поляков Д.Г., КТФ, гр. Р-11');
```

```
 WriteLn; { Вывод пустой строки }
WriteLn('Введите точность вычисления суммы');
Read(eps); { Ввод точности вычислений }
n:=0;  { Присвоение начального значения аргументу члена ряда }
repeat { Цикл с постусловием }
  x:=1; { Присвоение начального значения рекуррентной формуле
    вычисления факториала }
  for i:=1 to n do x:=x*i;
    k:=k+(n+1)/(2*x); { Накопление суммы членов ряда }
  n:=n+1; { Приращение аргумента }
until (n+1)/(2*x) < eps; { Выход из цикла при достижении заданной точности }
WriteLn('Выводится результат вычисляемого выражения');
WriteLn(k:4:5); { Форматный вывод }
ReadKey; { Задержка экрана }
end.
```

Пример программы на СИ: Программа вычисляет бесконечную сумму членов сходящегося ряда, причем расчет ведется от заданного n до тех пор, пока очередное значение выражения не станет меньше определенного числа a (точности расчета), вводимого пользователем с клавиатуры.

```
// Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами

void main(void) // Главная функция (основной блок программы)
{
 // Раздел объявления локальных переменных
  int n, i; // i-переменная счетчика цикла
  float s, ch, a, f; // s-сумма членов сходящегося ряда ch, f-факториал
  clrscr(); // Очистка экрана
  printf("Составитель:  Грибков Алексей Николаевич");
  printf("\n\t\tФакультет КТФ, группа Р-11");
  printf("\n\nВведите точность расчета (число a): ");
  scanf("%f", &a); // Ввод точности расчета
  n=1; // Начальное значение аргумента для вычисления члена ряда
  ch=1; // Начальное значение члена ряда
  s=0; // Присвоение начального значения рекуррентного выражения
    // накопления суммы
 while(ch>a) // Цикл выполняется до тех пор, пока очередное значение
  {       // выражения больше заданной точности
    f=1; // Присвоение начального значения рекуррентного выражения
      // вычисления факториала
    for(i=1; i<=n; i++) f*=i; // Вычисление факториала
    ch=(n+1)/(2*f); // Вычисление выражения
    s+=ch; // Накопление суммы
    n++; // Переход к следующему члену ряда
  }
 printf("Сумма равна s=%f", s); // Вывод полученного значения
                  // суммы на экран
 getch(); // Задержка экрана до нажатия любой клавиши
}
```

ПРОИЗВЕДЕНИЕ

Пример программы на СИ: Программа вычисляет произведение чисел ряда:

```
// Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами
#include<math.h> // Библиотека математических функций

void main(void) // Главная функция (основной блок программы)
{
 // Раздел объявления локальных переменных
  int n,i; // Переменные счетчика цикла
  float p,ch,m,f; // p-произведение членов ряда ch, f-факториал
  clrscr(); // Очистка экрана
  printf("Составитель:  Грибков Алексей Николаевич");
  printf("\n\t\tФакультет КТФ, группа Р-11");
  printf("\n\nВведите конец интервала (число m): ");
  scanf("%f",&m); // Ввод верхней границы произведения
```

```
    p=1; // Присвоение начального значения рекуррентной формуле
      // накопления произведения
    for(n=1;n<=m;n++) // Цикл выполняется до тех пор, пока n<m
    {
      f=1; // Присвоение начального значения рекуррентной формуле
        // вычисления факториала
        for(i=1;i<=n;i++) f*=i; // Вычисление факториала
      ch=(sqrt(f)+8*n)/(3*n-2); // Вычисление члена ряда
      p*=ch; // Вычисление произведения
    }
   printf("Произведение равно p=%f", p); // Вывод полученного значения
                                // произведения на экран
   getch(); // Задержка экрана до нажатия любой клавиши
}
```

СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

1 Цель лабораторной работы.
2 Блок-схема алгоритма программы.
3 Текст программы.
4 Результат расчета.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Что называется циклом?
2 Какие операторы называются начальными и конечными операторами цикла?
3 Что называется областью цикла?
4 Может ли параметр цикла быть а) переменной с индексом, б) простой переменной вещественного типа?
5 Может ли параметр цикла принимать нулевое и отрицательное значения?
6 В каком случае можно опустить значения приращения параметра цикла в заголовке цикла?
7 Какие операторы не могут быть конечными операторами цикла?
8 Каким образом размещаются элементы многомерных массивов в памяти ЭВМ?
9 Правила построения сложных циклов.

Литература [2, 3].

Лабораторная работа 4

СОСТАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОГРАММ, РАБОТАЮЩИХ С ИСПОЛЬЗОВАНИЕМ ОДНОМЕРНЫХ И ДВУМЕРНЫХ МАССИВОВ

Цель работы: получение практических навыков программирования алгоритмов обработки массивов.

ЗАДАНИЕ

1 Изучить особенности программирования алгоритмов работы с массивами на алгоритмических языках СИ и Паскаль.
2 Запрограммировать полученный алгоритм.
3 Ввести текст программы в компьютер.
4 Выполнить расчет по программе.

Для конкретного варианта составить массив исходных данных. Написать программу, реализующую заданный алгоритм обработки массива. Входную информацию и результаты расчета вывести на экран, снабдив их соответствующими заголовками.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Массивом называют упорядоченную последовательность величин, обозначенных одним именем. В отличие от переменной, которая занимает ячейку памяти ЭВМ, под массив отводится столько ячеек, сколько элементов в нем содержится. Элементы массива в памяти располагаются по строкам в СИ и в Паскале. Имя массива образуется в соответствии с общими правилами образования имен для используемого языка [3].

Отдельные элементы, образующие массив, называются элементами массива. К каждому элементу массива можно обратиться, указав имя массива и индексы, которые указывают положение массива. Таким образом, переменная с индексами является элементом массива.

Количество индексов, используемых для обозначения массива, зависит от размерности массива. Максимальная величина размерности равна семи. Индексы элементов массива записываются в скобках имени массива.

Тип массива должен соответствовать типу данных, которые образуют массив.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1 Получить вариант задания

ДЕЙСТВИЯ НАД ОДНОМЕРНЫМИ МАССИВАМИ

1) Найти сумму четных элементов массива.
2) Найти сумму положительных элементов массива.
3) Найти произведение элементов массива с четными индексами.
4) Найти количество нулевых и количество отрицательных элементов массива.
5) Найти сумму всех элементов массива, которые без остатка делятся на "6".
6) Найти количество положительных элементов массива, не превышающих числа "7".
7) Заменить отрицательные элементы нулями.
8) Увеличить все положительные элементы массива на единицу.
9) Найти произведение ненулевых элементов массива.
10) Увеличить элементы массива с четными индексами на "1", а элементы с нечетными индексами на "2".
11) Найти сумму отрицательных элементов массива и произведение положительных элементов.
12) Найти среднее арифметическое и среднее геометрическое значения элементов массива.
13) Указать номера элементов массива, которые без остатка делятся на "3" и на "5".
14) Указать номера элементов массива, которые больше "2", но меньше "5".
15) Указать, какой процент от общего количества составляют положительные элементы массива.
16) Сделать все положительные элементы массива отрицательными, вывести их индексы.

ОПЕРАЦИИ С ДВУМЕРНЫМИ МАССИВАМИ

1) Найти максимальный элемент среди элементов, находящихся выше главной диагонали квадратной матрицы. Указать его индексы.
2) Найти максимальный и минимальный элементы среди элементов, расположенных на побочной диагонали квадратной матрицы.
3) Поменять местами в матрице минимальный и максимальный элементы.
4) Найти в каждой строке матрицы минимальный элемент. Выделить максимальное из полученных значений.
5) Вычислить, в какой строке матрицы сумма элементов максимальная, а в какой – минимальная.
6) Найти в квадратной матрице сумму элементов, расположенных ниже побочной диагонали.
7) Найти в матрице максимальный и минимальный элементы, поменять местами строки, в которых они расположены.
8) Найти на главной диагонали квадратной матрицы максимальный и минимальный элементы. Поменять местами строки, в которых они расположены.
9) Найти для каждой строки матрицы сумму максимального и минимального элементов.
10) В строке матрицы с максимальным элементом обнулить все элементы, кроме максимального.
11) Для столбцов матрицы с четными номерами найти максимальный элемент, для столбцов с нечетными – минимальный.
12) Сформировать одномерный массив, элементы которого находятся суммированием элементов строк матрицы, в которых находятся максимальный и минимальный элементы.
13) Среди элементов квадратной матрицы, расположенных ниже побочной диагонали, найти минимальный и максимальный элементы.
14) Найти в каждой строке матрицы минимальный среди положительных элементов.
15) Поменять местами в матрице максимальный и минимальный положительные элементы.
16) Обнулить столбцы матрицы, в которых находятся максимальный и минимальный элементы.

2 Составить программы вычисления значений арифметических выражений на языках C++ и Паскаль. Обозначить в текстах программы порядок выполнения арифметических операций.

3 Выполнить расчет по программе.

Пример программы на Паскале:

```
{ Найти сумму четных элементов массива. }

uses crt; { Подключение библиотеки ввода-вывода }
const n = 10; { Размерность массива }
var i, s: Integer; { i-счетчик цикла,
      s-переменная накопления суммы четных элементов массива }
 a: Array [1..n] of Integer; { Объявление одномерного массива }
begin
  ClrScr; { Очистка экрана }
  WriteLn('Выполнил Поляков Д.Г., КТФ, Р-11 ');
  WriteLn; { Вывод пустой строки }
  for i := 1 to n do
  begin
    Write('Введите ',i,'-й элемент массива ');
    ReadLn(a[i]); { Ввод элемента массива }
  end;
 for i := 1 to n do
   Write(a[i]:2,' '); { Форматный вывод элементов массива }
 WriteLn; { Вывод пустой строки }
 s := 0; { Инициализация }
 i := 1; { переменных }
```

```
repeat
  if a[i] mod 2 =0 then
     s := s+a[i]; { Накопление суммы четных элементов }
  i := i+1; { Увеличение счетчика цикла }
until i > n; { Проверяется условие выхода за пределы массива }
WriteLn('Выводится сумма четных элементов массива: ',s);
ReadKey; { Задержка экрана }
end.
```

Пример программы на СИ: Программа вычисляет сумму четных элементов одномерного массива.

```
// Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами
#define N 6 // Макроопределение (размерность массива)

void main(void) // Главная функция (основной блок программы)
{
 // Раздел объявления локальных переменных
  int mas[N]; // Одномерный массив из N элементов
  int i,s;  // i-переменная счетчика цикла, s-сумма четных элементов
  clrscr(); // Очистка экрана
  printf("Составитель:  Грибков Алексей Николаевич");
  printf("\n\t\tФакультет КТФ, группа Р-11\n\n");
  for(i=0;i<N;i++)
 {
    printf("Введите %i элемент массива: ", i);
    scanf("%i", &mas[i]); // Ввод элементов массива
 }
 s=0; // Присвоение начального значения переменной s
 for(i=0;i<N;i++)
 {
   if(mas[i]%2==0) // Проверка на четность (деление на два без остатка)
   s+=mas[i]; // Накопление суммы
 }
 // Вывод полученного значения суммы на экран
 printf("\nСумма четных элементов равна s=%i", s);
 getch(); // Задержка экрана до нажатия любой клавиши
}
```

```
{ Задание 1. Найти максимальный элемент среди элементов,
 находящихся выше главной диагонали квадратной матрицы.
 Указать его индексы. }
uses crt; { Подключение библиотеки ввода-вывода }
const n = 3; { Задается размерность матрицы }
var i, j, max, imax, jmax: Integer;
  a: Array [1..n,1..n] of Integer;
begin
  ClrScr;  { Очистка экрана }
  WriteLn('Составил Поляков Д.Г., КТФ, Р-11');
  WriteLn; { Вывод пустой строки }
 for i := 1 to n do { Во вложенном цикле }
 begin      { происходит заполнение матрицы }
   for j := 1 to n do
   begin
     WriteLn('Введите ',i,'.',j,'-й элемент матрицы');
     Read(a[i,j]); { Ввод элементов матрицы }
   end;
   WriteLn; { Вывод пустой строки }
 end;
 for i := 1 to n do { Во вложенном цикле }
 begin      { происходит вывод матрицы }
   for j := 1 to n do Write(a[i,j]:3,' ');
     WriteLn; { Вывод пустой строки }
 end;
 max := a[2,1];
 for i := 1 to n do  {Производится поиск макс. элемента и его координат }
 begin       { согласно заданию. }
    for j := 1 to n do
       if (j-i) >= 1 then
          if a[i,j] > max then
```

```
            begin
               max := a[i,j];
               imax := i;
               jmax := j;
        end;
     end;
     WriteLn('Выводится max: ',max); { Выводятся макс. элемент и его координаты }
     WriteLn('Выводится строка: ',imax); { согласно заданию. }
     WriteLn('Выводится столбец: ',jmax);
     ReadKey;  { Производится задержка экрана }
end.
```

Пример программы на СИ: Программа вычисляет максимальный элемент среди элементов, находящихся выше главной диагонали квадратной матрицы.

```
// Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами
#define N 3 // Макроопределение (размерность массива)

void main(void) // Главная функция (основной блок программы)
{
 // Раздел объявления локальных переменных
 int mas[N][N]; // Двумерный массив (квадратная матрица NxN)
 int imax,jmax,i,j,max; // i, j-переменные счетчиков цикла
                        // imax, jmax-индексы максимального элемента
                        // матрицы, находящегося выше главной диагонали
                        // max-зачение максимального элемента
 clrscr(); // Очистка экрана
 printf("Составитель:  Грибков Алексей Николаевич");
 printf("\n\t\tФакультет КТФ, группа Р-11\n\n");
 for(i=0;i<N;i++)
   for(j=0;j<N;j++)
  {
     printf("Введите [%i][%i] элемент массива: ",i,j);
     scanf("%i",&mas[i][j]); // Ввод элементов массива
  }
 puts("\nВы ввели следующую матрицу:");
 // Вывод матрицы на экран
 for(i=0;i<N;i++)
    for(j=0;j<N;j++)
    {
       if(j==N-1) printf(" %i\n",mas[i][j]);
       else printf(" %i ",mas[i][j]);
    }
 // Нахождение максимального элемента матрицы
 max=mas[0][0]; imax=0; jmax=0;
 for(i=0;i<N;i++)
    for(j=0;j<N;j++)
  {
   if(j>=i) if(mas[i][j]>max)
   {
      max=mas[i][j];
      imax=i;
      jmax=j;
   }
}
   // Вывод полученного значения на экран
   printf("\nМаксимальный элемент-mas[%i][%i]=%i", imax, jmax, max);
   getch(); // Задержка экрана до нажатия любой клавиши
}
```

СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

1 Цель лабораторной работы.
2 Блок-схема алгоритма программы.
3 Текст программы.
4 Результат расчета.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Как образуется имя массива?
2 Как можно обратиться к отдельному элементу массива?

3 Каковы правила записи индексов?
4 Как определяют индексы в двумерном массиве?
5 Каким может быть тип массива?
6 Как располагаются в памяти ЭВМ элементы массива?
7 Каким образом можно присвоить начальные значения элементам массива?

Литература (2, 3).

Лабораторная работа 5

СОСТАВЛЕНИЕ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ПРОЦЕДУР И ФУНКЦИЙ

Цель работы: получение навыков в составлении программ с использованием процедур и функций.

ЗАДАНИЕ

1 Составить блок-схему алгоритма с использованием процедур и функций.
2 Изучить особенности программирования алгоритмов на алгоритмических языках СИ и Паскаль.
3 Запрограммировать полученный алгоритм.
4 Ввести текст программы в компьютер.
5 Выполнить расчет по программе.

Для выражений из лабораторной работы 2 вычисление 4-го элемента выражения оформить как функцию.
В задании к лабораторной работе 2 ввод исходных данных оформить в виде процедуры.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Функции – это фундаментальные логические элементы, служащие для выполнения действий, связанных с решением поставленной задачи.

Следует различать описание и представление функции. Описание делает возможным доступ к функции (помещает ее в область видимости), про которую известно что она внешняя. Представление задает действия, выполняемые функцией при ее вызове.

Представление функции включает:

* Тип значения, возвращаемого функцией (если значение не возвращается, то тип функции – void).
*Число и тип формальных параметров.
*Код (тело) функции, который должен быть выполнен при вызове функций.
* Указания о видимости функции вне файла, где она задается.
- Локальные переменные, которые могут маскировать глобальные переменные.

В (Турбо) СИ все параметры функции, за исключением параметров типа указатель и массивов, передаются по значению. При передаче параметра по значению в функции создается локальная копия, что приводит к увеличению объема используемой памяти. При вызове функции в сетке отводится память для локальных копий параметров, а при выходе из функции эта память освобождается. Рассмотренный способ использования памяти не только требует дополнительного пространства, но и отнимает часть времени счета.

При задании массива в качестве параметра функции передается адрес первого элемента массива. Если в теле функции заменяется значение элементов массива, то изменяется непосредственно сам передаваемый массив.

Если в описании функции задано, что параметр передается по ссылке (т.е. он описан как указатель на тип), то в качестве параметра при вызове функции передается адрес переменной.

Пример передачи массива в функцию как параметр

```
#include<stdio.h>
int data[2][2]={-12,14,-10,16};
main()
{
  extern void modify(int *a,int size);
  int i;
//modify(data,4);
 for (i=0;i<4;i++)
     printf(" \ndata[%d]=%d\n",i, *(data+1));
}
void modify(int *a,int size)
{
   int i;
   for (i=0;i<size;i++)
       a[i]++;
}
```

Пример передачи массива в функцию по значению

```
#include<stdio.h>
#define size 5
```

```
void main(void)
{
   int mas[size];
   int i,j;
   printf("\n max=%d",max(mas,size));
   printf("\n min=%d", min(mas,size));
}
 int max(int data[],int s)
{
   int maximum=data[0];
   int i,j;
   for (i=0;i<s;i++)
      if(data[i]>maximum)  maximum=data[i];
   return(maximum);
}
int min(int data[],int s)
{
   int minimum=data[0];
   int i,j;
   for (i=0;i<s;i++)
     if(data[i]<=minimum) minimum=data[i];
   return(minimum);
}
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 Получить вариант задания.

2 Составить программы вычисления значений арифметических выражений на языках СИ и Паскаль. Обозначить в текстах программ порядок выполнения арифметических операций.

3 Выполнить расчет по программе.

Пример программы на Паскале:

```
{ Составление простейших вычислительных }
 { программ циклической структуры }
 { с помощью функций }
uses crt; { Подключается библиотека ввода-вывода }
var m:Integer; { Наибольшее значение аргумента }

{ Функция для подсчета факториала }
Function Fact(x:Integer):Integer;
 var k, i : Integer; { k-переменная накопления значения факториала,
          i–счетчик цикла }
 begin
    k := 1; { Присвоение начального значения рекуррентной формуле
      вычисления факториала }
    for i := 1 to x do k:=k*i; { Вычисление факториала}
    Fact := k; { Возврат вычисленного значения факториала }
 end;{Function Fact}

{В данной функции производится расчет произведения ряда множителей}
Function Raschet(m:Integer):Real;
 var n, i : Integer; { n–номер члена ряда, i–счетчик цикла }
   k : Real; { Переменная накопления произведения членов ряда }
   begin
     n := 1; { Присвоение начального значения аргументу члена ряда }
     k := 1; { Присвоение начального значения рекуррентной формуле
       произведения членов ряда }
   repeat { Цикл с постусловием}
     for i:=1 to n do
      k := k * (Sqrt(Fact(n))+8*n)/(3*n–2); { Накопление произведения
                  членов ряда }
     n := n+1; { Приращение аргумента }
   until n > m; { Сравнение аргумента с верхней границей диапазона }
     Raschet := k;
 end;{Function Raschet}

{ Основной блок программы }
begin
 ClrScr;{ Производится очистка экрана }
 WriteLn('Выполнил Поляков Д.Г., КТФ, Р-11 ');
 WriteLn; {Вывод пустой строки}
```

```
 WriteLn('Введите количество множителей ');
 ReadLn(m); {Вводится верхняя граница диапазона изменения аргумента}
 WriteLn('Выводится результат вычисляемого выражения');
 WriteLn(Raschet(m):4:5); { Форматный вывод }
 ReadKey; { Задержка экрана }
end.
```

Пример программы на СИ: Программа вычисляет бесконечную сумму членов сходящегося ряда, причем расчет ведется от заданного n до тех пор, пока очередное значение выражения не станет меньше определенного числа a (точности расчета), вводимого пользователем с клавиатуры.

```
// Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами

// Раздел объявления глобальных переменных
int n, a; // a–точность расчета

// Прототипы функций
float factorial(int f) // Функция вычисления факториала
float raschet(float a) // Функция вычисления суммы членов ряда

// Функция вычисления факториала
// Возвращает факториал передаваемого в нее числа
float factorial(int f)
{
 // Раздел объявления локальных переменных
   int i; // Переменная счетчика цикла
   f=1; // Присвоение начального значения рекуррентного выражения
     // вычисления факториала
   for(i=1; i<=n; i++) f*=i; // Вычисление значения факториала
   return(f); // Возвращение значения факториала
}

// Функция расчета суммы членов ряда
float raschet(float a)
{
 // Раздел объявления локальных переменных
   float s, ch; // s-сумма членов сходящегося ряда ch
   n=1; // Начальное значение аргумента для вычисления члена ряда
   ch=1; // Начальное значение члена ряда
   s=0; // Присвоение начального значения рекуррентного выражения
     // накопления суммы
 while(ch>a) // Цикл выполняется до тех пор, пока очередное значение
  {       // выражения больше заданной точности
     ch=(n+1)/(2*factorial(n)); // Вычисление выражения
     s+=ch; // Накопление суммы
     n++; // Переход к следующему члену ряда
  }
 return(s); // Возвращение значения суммы
}

void main(void) // Главная функция (основной блок программы)
{
   clrscr(); // Очистка экрана
   printf("Составитель:  Грибков Алексей Николаевич");
   printf("\n\t\tФакультет КТФ, группа Р-11");
   printf("\n\nВведите точность расчета (число a): ");
   scanf("%f", &a); // Ввод точности расчета
   printf("Сумма равна s=%f", raschet(a)); // Вывод полученного значения
                          // суммы на экран
   getch(); // Задержка экрана до нажатия любой клавиши
}
```

СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

1 Цель лабораторной работы.
2 Блок-схема алгоритма программы.
3 Текст программы.
4 Результат расчета.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Какая разница между процедурой и функцией?
2 Чем различаются описание и представление функций?
3 Содержание отчета по лабораторной работе.

Лабораторная работа 6

ФАЙЛЫ С ПОСЛЕДОВАТЕЛЬНЫМ И ПРЯМЫМ ДОСТУПОМ

Цель работы: Получение навыков в организации файлов на внешних носителях и использование их для обработки информации.

ЗАДАНИЕ

1 Организовать ввод данных с клавиатуры и запись их в файл(ы) с прямым доступом.

2 Прочитать данные из файла(ов), произвести необходимые вычисления и результаты записать в файл с последовательным доступом.

3 Вывести результаты работы программы на экран, прочитав файл результатов.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Язык (Турбо) СИ не содержит средств поддержки ввода/вывода. Каждая реализация СИ должна сопровождаться библиотечными и макро, обслуживающими ввод/вывод. По мере развития СИ функции ввода/вывода становились стандартизированными. Такая стандартизация обеспечила высокий уровень мобильности программы на СИ.

Термин поток происходит из представления о последовательной структуре информационных записей [3]. Состав потока задается структурой FILE, описание которой задается в файле stdio.h.

Символ EOF определяется следующим образом:

#define EOF(-1)

Этот символ в операциях ввода/вывода служит для обозначения и проверки конца файла.

Функция fopen используется для открытия потока (файла). Интерфейс с функцией fopen описывается следующим образом:

FILE fopen(char filename, char type);

В качестве первого параметра функции должно передаваться правильное имя файла.

Второй параметр определяет тип открываемого файла.

Допустимы следующие типы файлов:

"r" Открыть уже существующий файл на ввод/

"w" Создать новый файл или очистить уже существующий файл и открыть его на вывод.

"a" Создать новый файл для вывода или осуществить вывод в конец уже существующего файла.

"r+" Открыть существующий файл для обновления, которое будет проводиться с начала файла.

"w+" Создать новый или открыть существующий файл для обновления его содержимого.

"a+" Создать новый файл или подстроиться в конец существующего файла для обновления его содержимого.

Функция flclose выглядит следующим образом:

int fclose (FILE *stream);

С помощью этой функции файл закрывается.

Функция fseek описывается следующим образом:

int fseek (FILE *stream, long offset, int wherefrom);

Эта функция служит для произвольного доступа к байтам, обычно внутри двоичных потоков.

Первый аргумент задает поток, к которому должен осуществляться прямой доступ.

Второй аргумент указывает число байт смещения от точки, определяемой третьим параметром функции.

Третий параметр указывает точку, от которой следует начинать отсчет смещения, заданного вторым аргументом.

Значение 0 – смещение от начала файла.

Значение 1 – смещение от текущей позиции файла.

Значение 2 – смещение от конца файла.

Пример:

```
#include<stdio.h>
#include<conio.h>
FILE *fff;
FILE *rrr;
FILE *ttt;
main()
{
  int mas1[]={1,-2,3};
  int mas2[]={-1,2,-3};
  int i, buf;
  fff=fopen("pol","w+");
  rrr=fopen("otr","w+");
  ttt=fopen("rez","w");
  clrscr();
  for(i=0;i<3;i++)
  {
    fwrite(&mas1[i],1,sizeof(int),fff);
```

```
        fwrite(&mas2[i],1,sizeof(int),rrr);
      }
    fclose(fff);
    fclose(rrr);
    fff=fopen("pol","rb");
    while(fread(&buf,1,sizeof(int),fff)!=0);
    {
       if(buf>0)
     {
        fwrite(&buf,1,sizeof(int),ttt);
       printf("\n%d", buf);
     }
    }
    rrr=fopen("otr","rb");
    while(fread(&buf,1,sizeof(int),rrr)!=0)
    {
        if(buf<0){ fwrite(&buf,1,sizeof(int),ttt);
         printf("\n%d", buf); }
    }
    fclose(fff);
    fclose(rrr);
    fclose(ttt);
}
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 Получить вариант задания:

1) Сформировать файл из некоторых чисел. Записать во второй файл количество положительных, отрицательных и нулевых элементов файла.

2) Сформировать два файла из некоторых чисел. Получить третий файл, в который сначала записать элементы второго файла, а затем первого.

3) Сформировать два файла из некоторых чисел. Записать в третий файл положительные элементы первого и второго файлов.

4) Сформировать файл из некоторых чисел. Записать во второй файл сначала положительные, а потом отрицательные элементы первого файла.

5) Сформировать два файла из некоторых чисел. Записать в третий файл положительные элементы первого файла и отрицательные элементы второго.

6) Сформировать два файла из некоторых чисел. Записать в третий файл нулевые элементы первого файла и четные элементы второго.

7) Сформировать файл из некоторых чисел. Записать во второй файл положительные элементы первого файла, а в третий – отрицательные элементы первого файла.

8) Сформировать два файла из некоторых чисел. Записать в третий файл положительные элементы второго файла и отрицательные элементы первого.

9) Сформировать два файла из некоторых чисел. Записать в третий файл минимальный элемент первого файла и максимальный элемент второго.

10) Сформировать файл из некоторых чисел. Записать во второй файл из первого максимальный отрицательный элемент, в третий – минимальный положительный.

11) Сформировать два файла из некоторых чисел. Определить, в каком файле больше положительных элементов и результат записать в третий файл.

12) Сформировать два файла из некоторых чисел. Записать в третий файл положительные элементы того файла, в котором находится максимальное отрицательное число.

13) Сформировать файл из некоторых чисел. Записать во второй файл элементы первого файла с четными номерами, в третий – с нечетными.

14) Сформировать файл из некоторых чисел. Записать во второй файл все положительные элементы первого, а затем абсолютные величины отрицательных элементов.

15) Сформировать два файла из некоторых чисел. Записать в третий файл два первых элемента первого файла и два последних элемента второго.

16) Сформировать два файла из некоторых чисел. Записать в третий файл последний положительный элемент первого файла и первый отрицательный второго.

2 Составить программы вычисления значений арифметических выражений на языках СИ и Паскаль. Обозначить в текстах программ порядок выполнения арифметических операций.

3 Выполнить расчет по программе.

Пример программы на Паскале:

```
{ Сформировать файл из некоторых чисел. }
{ Записать во второй файл количество положительных, }
{ отрицательных и нулевых элементов файла. }
uses CRT; { Подключение библиотеки ввода-вывода }
const n=10; { Максимальный размер массива }
```

```
{ Раздел объявления переменных }
var
 File1, File2: FILE of Integer; { Переменные файлового типа }
 Road1, Road2: String[14]; { Строки для хранения имен файлов }
 a : Array[1..n] of Integer; { Массив для хранения введенных чисел }
 i, pol, otr, nul: Integer; { i-счетчик цикла }
{ pol-количество положительных элементов массива}
{ otr количество отрицательных элементов массива}
{ nul количество нулевых элементов массива}
{ Основной блок программмы }
begin
 ClrScr;       { Очистка экрана }
 WriteLn('Выполнил Поляков Д.Г., КТФ, гр. Р-11');
 Road1:='Test1.dat'; { Задание имен файлов }
 Road2:='Test2.dat';
 Assign(File1,Road1); { Связь файловой переменной с внешним файлом Road1}
 Rewrite(File1);   { Создание и открытие файла Test1.dat }
 Assign(File2,Road2); { Связь файловой переменной с внешним файлом Road2}
 Rewrite(File2);   { Создание и открытие файла Test2.dat }
 pol:=0; { Инициализация переменных }
 otr:=0;
 nul:=0;
 for i:=1 to n do
 begin
   WriteLn('Введите ',i,' элемент массива'); { Запрос у пользователя
                элементов массива}
   Read(a[i]);    { Ввод элементов массива }
   Write(File1,a[i]); { Запись их в файл Test1.dat }
 end;
 Close(File1);    { Закрытие файла Test1.dat }
 WriteLn;  { Пропуск строки }
 WriteLn('_____________________________________');
 Reset(File1); { открытие файла Test1.dat для проверки правильности записи }
 while not eof(File1) do { Чтение файла до тех пор, пока указатель }
          { текущей компоненты файла находится перед }
          { последней компонентой файла }
  begin
   Read(File1,a[i]);  { Считывание чисел из файла }
   Write(a[i]:2); { Распечатка считываемых чисел }
   if a[i]>0 then pol:=pol+1; { Подсчет числа положительных, отрицательных}
   if a[i]<0 then otr:=otr+1; { и нулевых элементов }
   if a[i]=0 then nul:=nul+1;
   Inc(i); { Приращение счетчика цикла }
 end;
 WriteLn; { Пропуск строки }
 WriteLn('_____________________________________');
 Write(File2,pol); { Запись числа положительных, отрицательных }
 Write(File2,otr); { и нулевых элементов в файл Test2.dat }
 Write(File2,nul);
 Close(File2);   { Закрытие файлов Test1.dat и Test2.dat }
 Close(File1);
 Assign(File2,Road2);  { Открытие файла Test2.dat }
 Reset(File2); { для проверки правильности записи и печати результатов}
 Read(File2,pol); WriteLn('Количество положительных: ',pol:3);
 Read(File2,otr); WriteLn('Количество отрицательных:',otr:3);
 Read(File2,nul); WriteLn('Количество нулевых:',nul:3);
 Close(File2);   { Закрытие файла Test2.dat }
 ReadKey; { Задержка экрана }
end..
```

Пример программы на СИ: Сформировать файл из некоторых чисел. Записать во второй файл количество положительных, отрицательных и нулевых элементов первого файла.

```
// Подключаемые библиотеки
#include<stdio.h> // Библиотека ввода-вывода
#include<conio.h> // Библиотека работы с окнами
#define N 6 // Макроопределение (размерность массива)

void main(void) // Главная функция (основной блок программы)
{
```

```
// Раздел объявления локальных переменных
  FILE *f1; // Указатель на первый файл
  FILE *f2; // Указатель на второй файл
  int mas[N]; // Массив, элементами которого заполняется первый файл
int i, buf, nul, otr, pol; // i – переменная счетчика цикла
                           // buf – переменная, в которую будут
                           // считыватся значения из первого файла
                           // pol, otr, nul – переменные количества
                           // положительных, отрицательных и нулевых
                           // элементов первого файла соответственно
clrscr(); // Очистка экрана
printf("Составитель:  Грибков Алексей Николаевич");
printf("\n\t\tФакультет КТФ, группа Р-11\n\n");
// Заполнение массива
for(i=0; i<N; i++)
{
  printf("Введите %i элемент массива: ", i);
  scanf("%i", &mas[i]); // Ввод элементов массива
}
// Создание в текущем каталоге файлов с именами file1 и file2
// и открытие их на запись
f1=fopen("file1","w+");
f2=fopen("file2","w+");
// Запись в файл с именем file1 элементов массива mas[N]
for(i=0; i<N; i++) fwrite(&mas[i],1,sizeof(int),f1);
fclose(f1); // Закрытие файла file1
pol=0; // Присвоение начального значения переменным
otr=0; // количества положительных, отрицательных и
nul=0; // нулевых элементов
// Открытие файла с именем file1 на чтение
fopen("file1","r+");
// Чтение из файла информации в переменную buf до тех пор,
// пока не наступит конец файла
while(fread(&buf,sizeof(int),1,f1)!=0)
 {
  // Если очередное значение переменной buf положительное, то
  // значение переменной pol увеличивается на 1
    if(buf>0) pol=pol+1;
  // Если очередное значение переменной buf отрицательное, то
  // значение переменной otr увеличивается на 1
    if(buf<0) otr=otr+1;
  // Если очередное значение переменной buf равно нулю, то
  // значение переменной nul увеличивается на 1
    if(buf==0) nul=nul+1;
 }
// Запись в файл file2 количества положительных, отрицательных
// и нулевых элементов первого файла file1
fprintf(f2,"В ПЕРВОМ ФАЙЛЕ СОДЕРЖИТСЯ:\n");
fprintf(f2,"\nположительных элементов: %i", pol);
fprintf(f2,"\nотрицательных элементов: %i", otr);
fprintf(f2,"\nнулевых элементов: %i", nul);
// Вывод сообщения о завершении работы программы
printf("РАБОТА С ФАЙЛАМИ ЗАВЕРШЕНА");
printf("\nНажмите любую клавишу");
getch(); // Задержка экрана до нажатия любой клавиши
}
```

СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

1 Цель лабораторной работы.
2 Блок-схема алгоритма программы.
3 Текст программы.
4 Результат расчета.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Что такое поток?
2 Какой конфигурацией задается поток?
3 Какая функция используется для открытия потока?
4 Какая функция используется для закрытия потока?

ЗАКЛЮЧЕНИЕ

В пособии, к сожалению, не нашел отражение ряд важных вопросов, таких как модульное и объектно-ориентированное программирование, препроцессорные средства, динамическое распределение памяти, нет описания стандартных библиотек и т.д. Все эти средства совершенно необходимы программисту, работающему на современном уровне, однако при написании пособия авторы руководствовались разумным, по их мнению, компромиссом между требованием полноты представляемого материала, с одной стороны, и громоздкостью изложения – с другой. Мы же оправдываем свои усилия хотя бы тем, что сделали первый шаг на пути освоения программирования, и надеемся, заинтересовали в дальнейшем изучении этих и других алгоритмических языков.

СПИСОК ЛИТЕРАТУРЫ

1 Вычислительная техника в инженерных и экономических расчетах : учебник для вузов. 2-е изд., перераб. и доп. / А.В. Петров и др. ; под ред. А.В. Петрова. – М. : Высшая школа, 1984. – 320 с.

2 Брябрин, В.М. Программное обеспечение персональных ЭВМ / В.М. Брябрин. – М. : Наука, 1988. – 272 с.

3 Каранчук, В.П. Основы применения ЭВМ : учебное пособие для вузов / В.П. Каранчук, И.Н. Сваровский, И.Д. Суздальницкий. – М. : Радио и связь, 1988. – 288 с.

4 Ноутон, П. Java : Пер с англ. / П. Ноутон, Г. Шилдт. – СПб. : Петербург, 2003. – 1070 с.

5 Ли, Т.Г. Управление процессами с помощью вычислительных машин. Моделирование и оптимизация / Т.Г. Ли, Г.Э. Адамс, У.М. Гейнз. – М. : Советское радио, 1972.

6 Уинер, Р. Язык Турбо СИ / Р. Уинер. – М. : Мир, 1991. – 380 с.