

Министерство образования и науки РФ  
Севастопольский государственный университет  
Кафедра информатики и управления в технических системах

## **ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ СИ**

**Методические указания**  
к выполнению лабораторных работ № 1-3  
по дисциплине  
"Алгоритмизация и программирование"  
для студентов очной и заочной форм обучения  
по направлениям подготовки  
27.03.04 "Управление в технических системах" и  
09.03.01 "Информатика и вычислительная техника"

Севастополь  
2017

УДК 004.43

**Основы языка программирования Си:** Методические указания к выполнению лабораторных работ по дисциплине «Алгоритмизации и программирование»/ Сост. А.А. Кабанов, В.В Захаров – Севастополь: Изд-во СевГУ, 2017. – 28 с.

Целью методических указаний является оказание помощи студентам при выполнении лабораторных работ, целью которых является приобретение навыков составления простейших алгоритмов, изучение основ языка Си, освоение базовых приемов работы в среде Dev-Cpp.

Методические указания предназначены для студентов дневной формы обучения по направлениям подготовки 27.03.04 "Управление в технических системах" и 09.03.01 "Информатика и вычислительная техника"

Методические указания рассмотрены и утверждены на заседании кафедры информатики и управления в технических системах (протокол № 7 от 30.09.2017 г.)

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

Рецензент

Кабанов А.А., канд. техн. наук, доцент

## СОДЕРЖАНИЕ

Введение .....	4
1. Лабораторная работа №1. Программирование линейных алгоритмов на языке Си.....	4
1.1. Цель работы .....	4
1.2. Задание на работу .....	4
1.3. Краткие теоретические сведения .....	7
1.3.1. Описание лабораторного стенда.....	7
1.3.2. Базовые типы данных языка Си.....	9
1.3.3. Некоторые функции стандартного ввода-вывода.....	10
1.3.4. Некоторые стандартные математические функции.....	11
1.4. Примеры программ.....	12
1.4.1. Пример программы для изучения базовых типов данных и средств форматированного ввода вывода .....	12
1.4.2. Пример программы с математическими функциями .....	13
1.5. Содержание отчета и порядок защиты работы .....	15
1.6. Контрольные вопросы.....	15
2. Лабораторная работа №2. Программирование разветвляющихся алгоритмов на языке Си.....	16
2.1. Цель работы .....	16
2.2. Задание на работу .....	16
2.3. Краткие теоретические сведения .....	18
2.4. Пример программы.....	18
2.5. Содержание отчета и порядок защиты работы .....	20
2.6. Контрольные вопросы.....	21
3. Лабораторная работа №3. Программирование циклических алгоритмов на языке Си.....	22
3.1. Цель работы .....	22
3.2. Задание на работу .....	22
3.2.1. Четные варианты заданий .....	22
3.2.2. Нечетные варианты заданий .....	22
3.3. Краткие теоретические сведения .....	24
3.4. Пример программы.....	26
3.5. Содержание отчета и порядок защиты работы .....	27
3.6. Контрольные вопросы.....	27
4. Библиографический список.....	27
Приложение А. Перечень тем блока №4.....	28

## ВВЕДЕНИЕ

Предлагаемый цикл лабораторных работ посвящен получению практических навыков программирования линейных, разветвляющихся и циклических алгоритмов. Цикл лабораторных работ ориентирован на применение языка программирования Си.

Лабораторные работы входят в блок №4 дисциплины «Алгоритмизации и программирование». Для успешной сдачи тем блока необходимо выполнить и защитить лабораторные работы *не позднее 8-ой недели семестра*.

Примерный перечень тем, входящих в блок №4, приведен в приложении А. Для самостоятельной подготовки рекомендуются учебники [1 – 5].

## 1. ЛАБОРАТОРНАЯ РАБОТА №1. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ НА ЯЗЫКЕ СИ

### 1.1. Цель работы

Целью лабораторной работы является исследование линейных алгоритмов на языке Си средствами среды разработки Dev-C++ или Borland C++ 3.1. Для достижения поставленной цели необходимо решить ряд задач: изучить базовые типы данных, а также получить практические навыки использования функций ввода-вывода и математических функций стандартной библиотеки языка Си.

### 1.2. Задание на работу

1) Работа выполняется в среде Dev-C++. Описание лабораторного стенда приведено в пункте 1.3.1 настоящих методических указаний.

2) При выполнении лабораторной работы необходимо составить две программы. *Первая программа* предназначена для изучения базовых типов данных и средств форматированного ввода вывода. Программа должна обеспечивать ввод с клавиатуры данных по варианту задания (см. таблицу 1.1). Далее программа должна выводить эти данные на экран в виде одной строки с комментариями.

*Вторая программа* предназначена для изучения математических функций библиотеки языка С. Программа должна осуществлять вычисление по формулам, указанным в таблице 1.1 в соответствии с номером варианта. Преобразуйте формулы с целью уменьшения количества операций при вычислениях. Упрощение возможно как за счет математических преобразований, так и за счет введения дополнительных переменных для сохранения значений выражений, неоднократно встречающихся в формуле.

Таблица 1.1 – Варианты заданий

№ вар.	Программа 1	Программа 2
1.	<b>Данные жильца гостиницы:</b> Фамилия жильца Занимаемый номер Дата поселения (день, месяц, год) Стоимость проживания	$y = \frac{e^{-a} + \frac{z+10^3}{\sin z}}{\cos \pi z + \ln b}, z =  b-15,1 $
2.	<b>Данные сотрудника предприятия:</b> Фамилия Инициалы Должность Год приема на работу Оклад	$a = \frac{e^{-tg \pi k} - \ln  x }{kx + 10^5}, k = \frac{y+11}{2}$
3.	<b>Расписание электрички:</b> Номер электрички Пункт назначения Стоимость билета Время прибытия (часы, минуты)	$k = \frac{\ln  i + 8 \cdot 10^{-2}  - \ln i}{\cos \pi + e^x}, i = \frac{m-1}{m+1}$
4.	<b>Описание товара:</b> Код товара Название Цена Количество	$n = e^{\frac{-\sin m + tg \frac{\pi r}{p}}{p}}, p = \ln \left  \frac{r^3 + 10^3}{r^3 - 1} \right $
5.	<b>Характеристики монитора:</b> Модель Производитель Диагональ (дюймы) Время отклика Цена	$d = \ln \frac{z^2 + 10^{-3}}{z^2 + 1,6 \cdot 10^{-2}}, z = e^{-k} ctg kx$
6.	<b>Данные об автовладельце:</b> Фамилия владельца Марка автомобиля Регистрационный номер Год выпуска	$m = \sin \arctg \frac{z}{2} - \sin \arctg \frac{\pi}{3}$ $z = \frac{ e^{-xy} + 10^{-3}  + e^{-xy}}{\pi + \ln xy}$
7.	<b>Описание книги:</b> Фамилия автора Название книги Год издания Жанр Цена	$z = tg^2 \frac{\pi m}{m + 10^{-3}}, m = e^{-c x  + \sqrt{cx}}$

Таблица 1.1 – Варианты заданий

№ вар.	Программа 1	Программа 2
8.	<b>Описание продуктового товара:</b> Название товара Производитель Цена Сорт Срок хранения	$t = \frac{\sqrt{\cos^2 \pi r + k \cdot 10^{-2}}}{\cos^2 \pi r + \left  \frac{m}{k} \right }, m = \ln \cos \pi r$
9.	<b>Описание учебной дисциплины:</b> Название дисциплины Курс Семестр Фамилия преподавателя Наличие экзамена	$m = \frac{\sqrt{\ln k + 10^3} - \sqrt{10^3 - \ln k}}{x^2 + 18x - 40}$ $k = e^{\pi x} \cos 0,01z$
10.	<b>Данные библиографического источника:</b> Название литературного источника Автор Год издания Категория (книга, статья)	$k = tg^2 z + ctg^2 z, z = \frac{e^{\pi x} - e^{-\pi x}}{10^3 + \sqrt{\ln m x}}$
11.	<b>Билет в кино:</b> Название фильма Время начала сеанса Ряд Место Цена билета	$b = \sqrt{\frac{\sin^2 n + \sin n^2}{ \sin n  + e^{-x}}}, n = \ln \frac{\pi}{kx - 1,6 \cdot 10^3}$
12.	<b>Характеристики фотоаппарата:</b> Модель Цена Разрешение (в мегапикселях) Объем карты памяти Вес	$m = 180 \arctg \frac{e^{5t} - e^{-5t}}{e^{zt} - e^{-zt}}, t = \sqrt{\frac{\ln  \pi r }{10^3 + rz}}$
13.	<b>Данные из телефонного справочника:</b> Фамилия Имя Отчество Номер телефона	$a = \cos^2 tg \left( \frac{1}{b} \right), b = \frac{x + \frac{y}{5 + \sqrt{x}}}{ y + x  + \sqrt[3]{x}}$
14.	<b>Описание фильмов:</b> Название Жанр Бюджет фильма (долл.) Дата выхода в прокат (день, месяц, год)	$a = e^{b-1} + tg^2 x, b = \frac{2 \cos(x - \pi/6)}{\frac{1}{x} + \sin^2 y}$

## 1.3. Краткие теоретические сведения

### 1.3.1. Описание лабораторного стенда

Лабораторный стенд представляет собой персональный компьютер с установленной на нем средой программирования Dev-C++. Схема лабораторного стенда дана на рис.1.1.

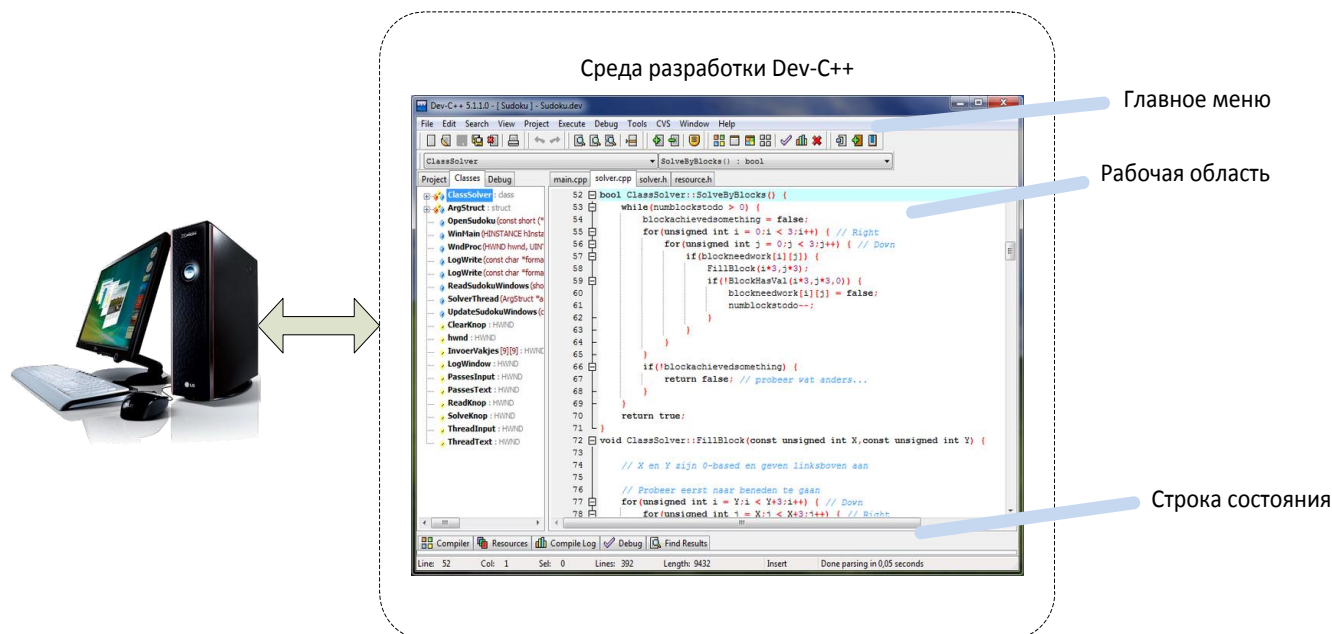


Рисунок 1.1 – Структура лабораторного стенда

Dev-C++ – это интегрированная среда для программирования на языках C и C++, работающая под управлением операционной системы Windows. Среда Dev-C++ распространяется свободно с исходными кодами (на Delphi) по лицензии GPL. Все эти компоненты объединены в единую интегрированную среду разработчика, с которой как раз и работает программист, создавая свои программы.

В дистрибутив входит компилятор MinGW. Основатель проекта – компания Bloodshed Software.

Рассмотрим некоторые особенности работы в этой среде:

1) Основные команды меню и соответствующие им «горячие клавиши» приведены в таблице 1.3.

2) Вывод русских букв.

В текстовом редакторе Dev-C++ используется кодировка символов Windows-1251– набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для всех русских версий Microsoft Windows. Однако, в консольном окне используется кодировка символов CP866. Поэтому, если не предпринять вспомогательных действий, русские буквы, набранные в текстовом редакторе среды, при выводе на экран будут отображаться некорректно.

Таблица 1.3 – Некоторые команды меню среды Dev-C++

Пункт меню		Сочетание клавиш	Назначение
File (Работа с файлами)	New → Source file	Ctrl-N	Создать новый исходный файл
	Open project or file	Ctrl-O	Открыть файл
	Save	Ctrl-S	Сохранить файл
	Save as...	Ctrl-F12	Сохранить файл под именем...
	Reopen		Открыть один из ранее открытых файлов
	Close	Ctrl-F4	Закрыть редактируемый файл
Execute (Выполнение программы)	Run	Ctrl+F10	Запустить программу без перекомпиляции
	Compile	Ctrl+F9	Компиляция
	Compile & Run	F9	Запустить программу, перекомпилировав исходный файл

Можно использовать следующий способ решения этой проблемы.

Шаг 1. В свойствах консольного окна измените шрифт с точечного на Lucida Console – точечные шрифты не поддерживают cp1251 (см. рисунок 1.1).

Шаг 2. Подключите заголовочный файл `<stdlib.h>` и добавьте в начало программы строку

```
system("chcp 1251");
```

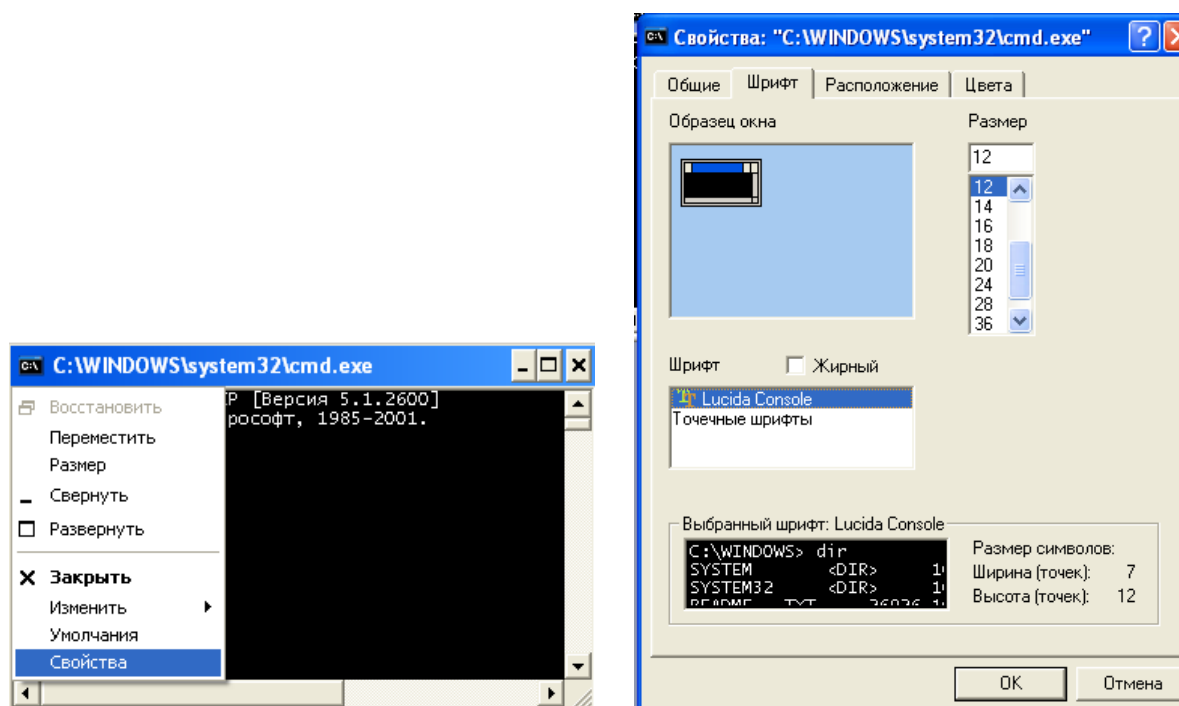


Рисунок 1.1 – Изменение шрифта консольного окна



Функция `system()` передает строку-аргумент операционной системе для выполнения. В данном случае вызывается команда MS DOS `chcp`, которая позволяет изменить текущую кодировку.

3) В среде Dev-C++ по окончании работы программы консольное окно закрывается. Чтобы предотвратить закрытие окна до того, как увидим последний вывод результатов работы программы, можно воспользоваться следующими способами.

**Способ 1.** Добавить в конец программы две строки:

```
fflush(stdin); /*очистка буфера входного потока */
getchar(); /* ожидание нажатия клавиши */
```

либо

```
getchar();
getchar();
```

Первой командой убираем из входного потока признак конца строки, внесенный при нажатии последнем клавиши Enter (если ранее осуществлялся ввод данных). Вторая команда заставляет программу ждать нажатия любой клавиши. И функция `fflush()`, и `getchar()` описаны в заголовочном файле `<stdio.h>`.

**Способ 2.** Подключить заголовочный файл `<stdlib.h>` и добавить в конец программы строку

```
system ("pause"); /* запуск команды MS-DOS: pause */
```

### 1.3.2. Базовые типы данных языка Си

Перечень базовых типов языка Си с их кратким описанием приведен в таблице 1.2.

Таблица 1.2 – Базовые и некоторые производные типы данных языка Си

Название типа	Пояснения	Диапазон значений
Short	Краткое целое число	-32768 ... 32767
unsigned short	Краткое целое число без знака	0 ... 65535
int	Целое число	-2147483648... 2147483647 для Dev-C++ -32768 . . . 32767 для для Borland C++ 3.1
unsigned int	Целое число	0...4294967295 для Dev-C++ 0 ... 65535 для для Borland C++ 3.1
long	Длинное целое число	-2147483648... 2147483647
unsigned long	Длинное целое число без знака	0...4294967295
char	Один символ	символы кода ASCII
char[ ]	Строка	
float	Число с плавающей точкой	$3.4 \cdot 10^{-38} \dots 3.4 \cdot 10^{+38}$
double	Число с плавающей точкой двойной точности	$1.7 \cdot 10^{-308} \dots 1.7 \cdot 10^{+308}$

### 1.3.3. Некоторые функции стандартного ввода-вывода

Чтобы использовать функции стандартного ввода-вывода, необходимо подключить к программе файл `stdio.h`.

`printf()` - форматированный вывод на экран:

```
int printf(char *format, <список вывода>);
```

Первый параметр является символьной строкой, которая задает спецификации формата. Остальные параметры - перечисление переменных и выражений, значения которых выводятся. Каждая спецификация формата имеет вид (параметры в квадратных скобках необязательны):

```
%[flags][width][.prec][F|N|h|l]type
```

где `type` – тип спецификации:

`d` или `i` – целое десятичное число со знаком;

`u` – десятичное число без знака;

`x` – целое 16-ричное число без знака;

`f` – число с плавающей точкой;

`e` – число в E-форме;

`g` – число с плавающей точкой или в E-форме;

`c` – один символ;

`s` – строка;

`%` – символ %;

`flags` – признак выравнивания:

`+` или пусто – выравнивание по правому краю;

`-` – выравнивание по левому краю;

`width` – целое число – общая ширина поля. Если это число начинается с цифры 0, вывод дополняется слева нулями до заданной ширины. В заданную ширину входят все символы вывода, включая знак, дробную часть и т.п.

`prec` – целое число, количество знаков после точки при выводе чисел с плавающей точкой;

`F` – соответствующий элемент списка вывода является дальним указателем;

`N` – соответствующий элемент списка вывода является близким указателем;

`l` – соответствующий элемент списка вывода является `long int` или `double`.

`scanf()` – форматированный ввод с клавиатуры:

```
int scanf(char *format, <список ввода>);
```

Первый параметр является символьной строкой, которая задает спецификации формата (см. функцию `printf()`). Остальные параметры - перечисление адресов переменных, в которые вводятся данные. В этом списке перед именами всех переменных, кроме тех, которые вводятся по спецификации типа `%s`, должен стоять символ `&`.

`putchar()` - вывод одного символа на экран:

```
int putchar(int ch);
```

Параметр функции – код символа, который выводится. При успешном выполнении функция возвращает этот же код, при неуспешном – EOF.

`getchar()` - ввод одного символа с клавиатуры:

```
int getchar(void);
```

Функция возвращает код введенного символа.

`puts()` - вывод строки символов на экран:

```
int puts(char *string);
```

Параметр функции – указатель на начало той строки, из которой выводятся данные. Функция возвращает количество выведенных символов.

`gets()` – ввод строки символов с клавиатуры:

```
char *gets(char *string);
```

Параметр функции – указатель на начало той строки, в которую вводятся данные. Функция возвращает тот же самый указатель.

### 1.3.4. Некоторые стандартные математические функции

Чтобы использовать стандартные математические функции стандартного ввода-вывода, необходимо подключить к программе файл `math.h`.

**abs** - абсолютное значение целого числа –  $|x|$

```
int abs(int x);
```

**labs** – абсолютные значения "длинного" целого числа –  $|x|$ :

```
long labs(long x);
```

**fabs** – абсолютное значение числа с плавающей точкой –  $|x|$ :

```
double fabs(double x);
```

**sqrt** – извлечение квадратного корня:

```
double sqrt(double x);
```

**pow** – возведение в степень  $x^y$  (ошибка области, если  $x = 0$  или  $y \leq 0$  или  $x < 0$  и  $y$  – не целое):

```
double pow(double x, double y);
```

**cos** – косинус –  $\cos x$  (здесь и далее  $x$  задается в радианах):

```
double cos(double x);
```

**sin** – синус –  $\sin x$ :

```
double sin(double x);
```

**tan** – тангенс –  $\tan x$ :

```
double tan(double x);
```

**acos** – арккосинус –  $\arccos x$ :

```
double cos(double x);
```

**asin** – арксинус –  $\arcsin x$ :

```
double sin(double x);
```

**atan** – арктангенс –  $\arctg x$ :

```
double atan(double x);
```

**atan2** – арктангенс –  $\arctg x/y$ :

```
double atan2(double x, double y);
```

**exp** – экспонента  $e^x$ :

```
double exp(double x);
```

**log** – натуральный логарифм –  $\ln x$ :

```
double log(double x);
```

**log10** – десятичный логарифм –  $\log_{10}x$ :

```
double log10(double x);
```

## 1.4. Примеры программ

В данном подразделе приведены примеры двух программ: первая программа демонстрирует работу с базовыми типами данных и средствами форматированного ввода вывода, вторая – с математическими функциями.

### 1.4.1. Пример программы для изучения базовых типов данных и средств форматированного ввода вывода

Рассматривается следующая постановка задачи. Написать программу, которая читала бы с клавиатуры следующие данные о студенте:

- фамилия;
- инициалы;
- номер зачетки;
- средний балл.

Программа должна выводить эти данные на экран в виде одной строки с комментариями.

Текст программы приведен ниже.

```
#include <stdio.h>
#include <stdlib.h>
int main (void)
{   char fam[15]; /* фамилия (до 14 символов) */
    char im, ot; /* инициалы (по одному символу) */
    unsigned int nzach; /*номер зачетки (целое без знака) */
```

```

float ball;      /* средний балл (вещественное число) */
/* смена кодировки для вывода русских букв */
system("chcp 1251");
/* ввод данных */
printf("Введите фамилию:\t");
scanf("%s", fam); /* нет операции взятия адреса &,
                  т.к. fam - массив символов */
printf("Введите первую букву имени:\t");
fflush(stdin); /* очистка буфера входного потока - уби-
               раем из входного потока признак конца
               строки, внесенный при нажатии клавиши
               enter. Если этого не сделать, в im будет
               занесен символ конца строки, а не буква.
               Другой способ: вызвать getchar(); */
im = getchar(); /* другой способ: scanf("%c",&im); */
printf("Введите первую букву отчества:\t");
fflush(stdin);
scanf("%c",&ot); /* или ot = getchar(); */
printf("Введите номер зачетки:\t");
scanf("%u",&nzach);
printf("Введите средний балл:\t");
scanf("%f",&ball);
/* вывод данных */
printf("Студент %s %c.%c. (номер зачетки %6u) имеет
средний балл %4.2f \n", fam, im, ot, nzach, ball);
/* ожидание нажатия клавиши */
fflush(stdin);
getchar();
return 0;

```

#### 1.4.2. Пример программы с математическими функциями

Постановка задачи: Вычислить значение  $a$  по формуле

$$a = \frac{\sqrt[3]{|y|}}{(x+1) + (x-1)} + \frac{|\sin x|}{x\sqrt{|y|}}$$

Предварительный расчет  $a$  для  $x = 1,241$ ,  $y = -0,879$  на калькуляторе дает:

$$\begin{aligned}
 z &= 0,879, \\
 \sin x &= 0,9461, \\
 a &= \frac{0,9579}{2,482} + \frac{0,9461}{0,9013} = 0,3859 + 1,0497 = 1,4357.
 \end{aligned}$$

Алгоритм программы будет иметь вид, показанный на рисунке 1.2.

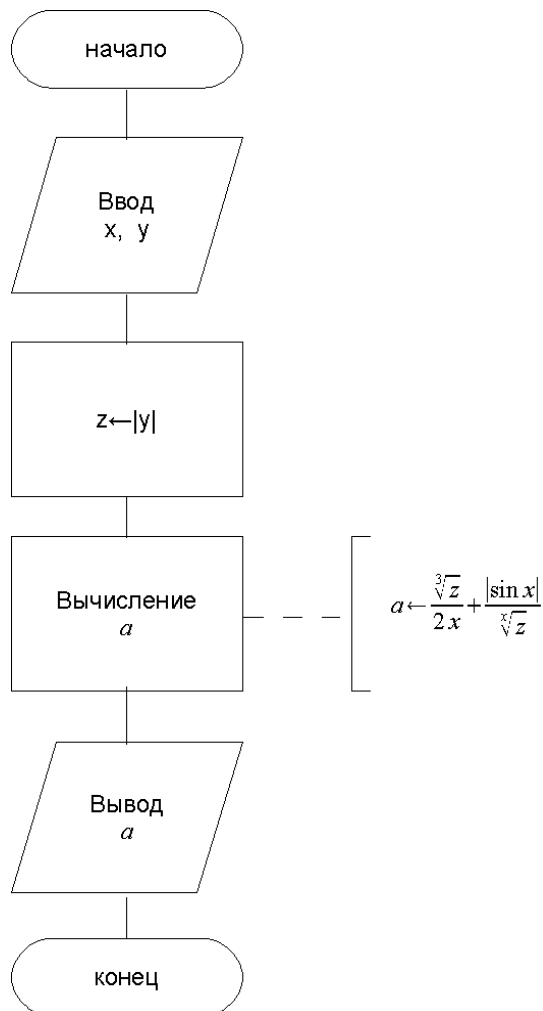


Рисунок 1.2 – Схема программы

Текст программы на языке **Си** будет выглядеть следующим образом:

```

#include <stdio.h>
#include <math.h>
int main (void)
{
    double x, y; /* входные данные */
    double a;    /* результат */
    double z;    /* вспомогательная переменная*/
    /* ввод исходных данных */
    printf("x= ");
    scanf("%lf",&x); /* для типа double к спецификатору фор-
    мата f добавляется l (как бы long float)*/
    printf("y= ");
    scanf("%lf",&y);

    printf("x= %lf",x);
    printf("y= %lf",y);
    /* вычисления */
    z = fabs(y);

```

```

    a = exp(log(z)/3)/2/x + fabs(sin(x))/exp(log(z)/x) ;
/* здесь нельзя использовать pow() для вычисления степени,
т.к. pow() вычисляет только целую степень */
/* вывод результата */
    printf("a= %6.3lf\n", a);
/* ожидание нажатия клавиши */
    system("pause"); }

```

## 1.5. Содержание отчета и порядок защиты работы

Выполнение и защита лабораторной работы производится каждым студентом индивидуально. Защита результатов лабораторной работы осуществляется при наличии работающей программы и полностью оформленного отчета.

Отчет должен включать в себя следующие разделы

- титульный лист;
- цель работы;
- постановка задачи. Этот раздел должен содержать вариант задания в соответствии с таблицей вариантов заданий;
- схемы программ;
- тексты программ на языке Си;
- результаты работы программ;
- выводы.

Защита работы состоит в следующем:

- представление работающих программ на компьютере;
- предъявление отчета, оформленного в соответствии с требованиями;
- ответы на вопросы преподавателя по теоретической и практической части работы. Примеры возможных вопросов приведены в подразделе 1.6.

## 1.6. Контрольные вопросы

- 1) Из каких частей состоит программа на Си?
- 2) Перечислите этапы создания исполняемой программы на языке Си.
- 3) Что такое директива препроцессора? Привести примеры директив препроцессора.
- 4) Какие типы данных называют базовыми, а какие производными?
- 5) Каковы диапазоны значений переменных типа int, типа unsigned int?
- 6) Какой объем оперативной памяти занимают данные типа long int?
- 7) Чем отличаются друг от друга типы float, double и long double?
- 8) Что является первым параметром функций printf() и scanf()? Для чего он предназначен?
- 9) Какие спецификации используют для вывода целых чисел?
- 10) Какую роль играют элементы спецификации [width] и [.prec] при выводе вещественных чисел?
- 11) В каких случаях ввода-вывода данных необходимо использовать модификаторы l и L?

## 2. ЛАБОРАТОРНАЯ РАБОТА №2. ИССЛЕДОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ НА ЯЗЫКЕ СИ

### 2.1. Цель работы

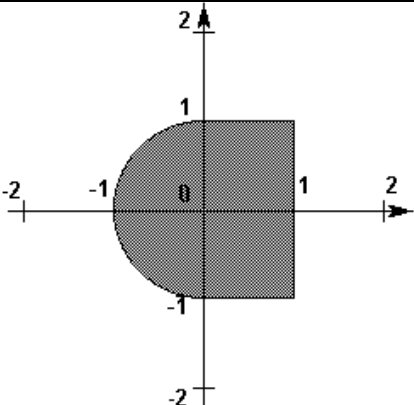
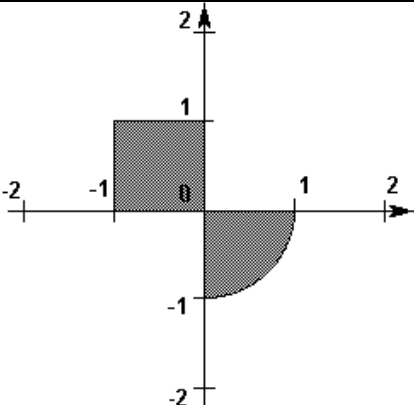
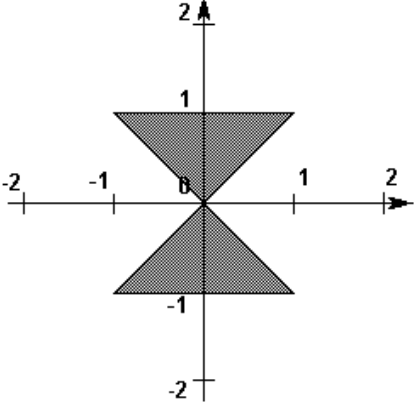
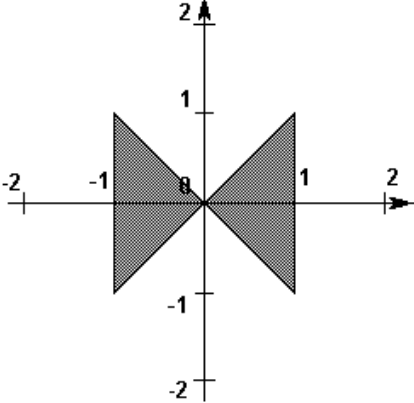
Целью данной работы является исследование разветвляющихся алгоритмов и их программирование с помощью условного оператора языка Си. Для достижения данной цели студентам необходимо: закрепить навыки программирования ввода и вывода информации, получить начальные навыки тестирования программ.

### 2.2. Задание на работу

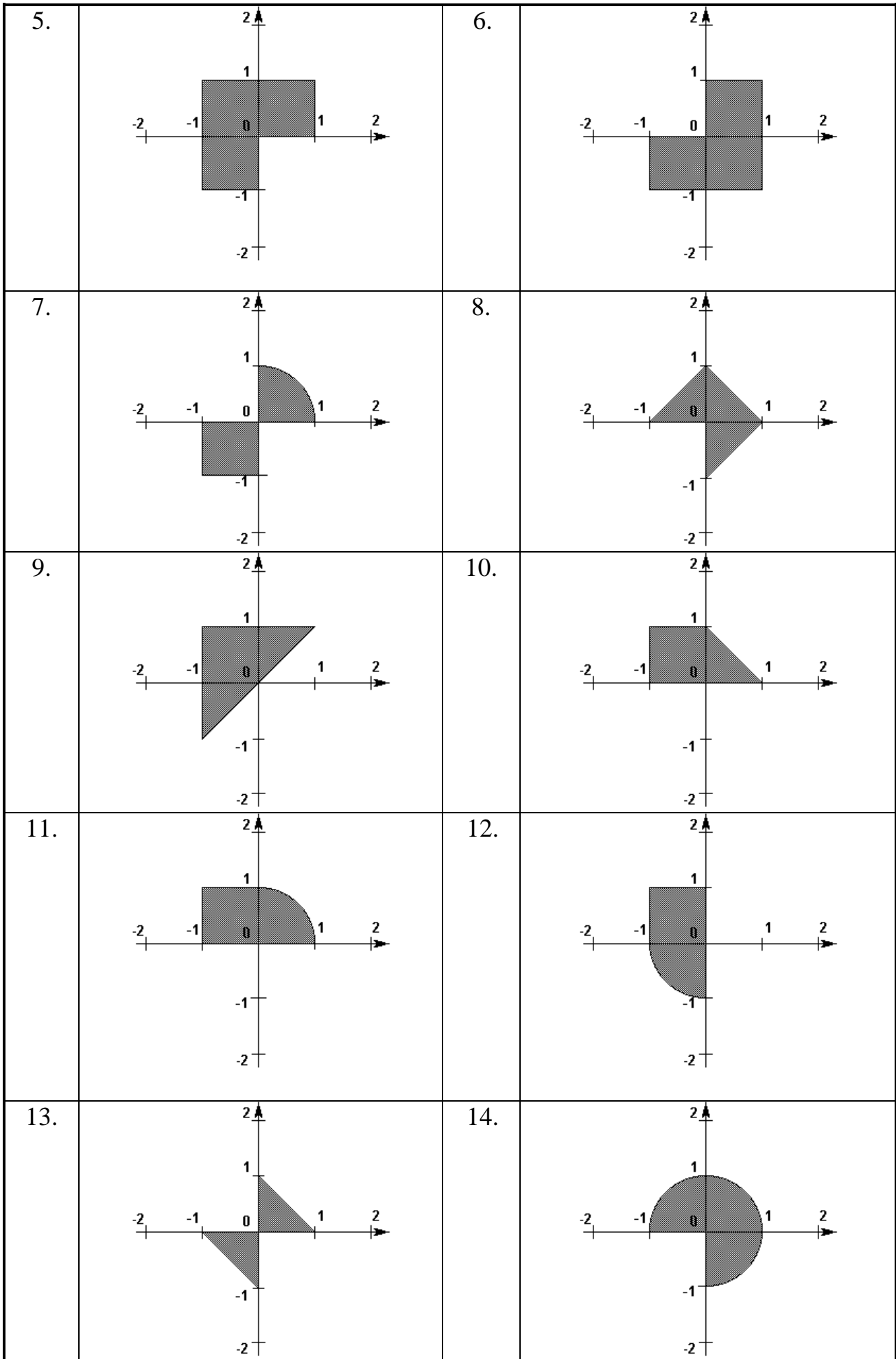
1) Работа выполняется в среде Dev-C++ . Описание лабораторного стенда приведено в методических указаниях к лабораторной работе № 1.

2) Дана заштрихованная область и точка с координатами  $(x, y)$ . Вид области выбирается из таблицы 2.1 в соответствии с номером варианта. Требуется написать программу, определяющую, попадает ли точка в область. Результат вывести в виде текстового сообщения.

Таблица 2.1 – Варианты заданий

№ варианта	Область	№ варианта	Область
1.		2.	
3.		4.	





### 2.3. Краткие теоретические сведения

Условный оператор `if` дает возможность выбрать один из двух возможных вариантов продолжения программы.

Синтаксис оператора `if` можно изобразить таким образом:

```
if (выражение) оператор1; [else оператор2;]
```

Выполнение оператора `if` начинается с вычисления выражения.

Далее выполнение осуществляется по следующей схеме:

- если выражение истинно (т.е. отлично от 0), то выполняется оператор1;
- если выражение ложно (т.е. равно 0), то выполняется оператор2;
- если выражение ложно и отсутствует оператор-2 (в квадратные скобки заключена необязательная конструкция), то выполняется следующий за `if` оператор.

Для формирования условий используются операции отношения и логические операции.

Операции отношения (см. таблицу 2.2) используются для сравнений.

Таблица 2.2 – Операции отношения языка Си

Обозначение	Назначение
<	Меньше
<=	меньше или равно
==	Равно
>=	больше или равно
>	Больше
!=	не равно

В языке Си имеются три логические операции, приведенные в таблице 2.3.

Таблица 2.3– Логические операции языка Си

Обозначение	Назначение	
&&	И	выражение1 && выражение2: истинно тогда и только тогда, когда оба выражения истинны
	ИЛИ	выражение1    выражение2: истинно, если какое-нибудь одно или оба выражения истинны
!	НЕ	!выражение: истинно, если выражение ложно, и наоборот

### 2.4. Пример программы

Рассмотрим задачу из [3]. Дана заштрихованная область, показанная на рисунке 2.1, и точка с координатами  $(x, y)$ . Написать программу, определяющую, попадает ли точка в область. Результат вывести в виде текстового сообщения.

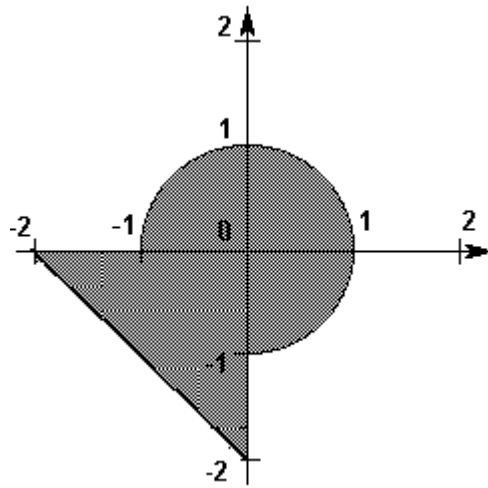


Рисунок 2.1 – Графически заданная область

Алгоритм программы изображен в виде схемы, показанной на рисунке 2.2.

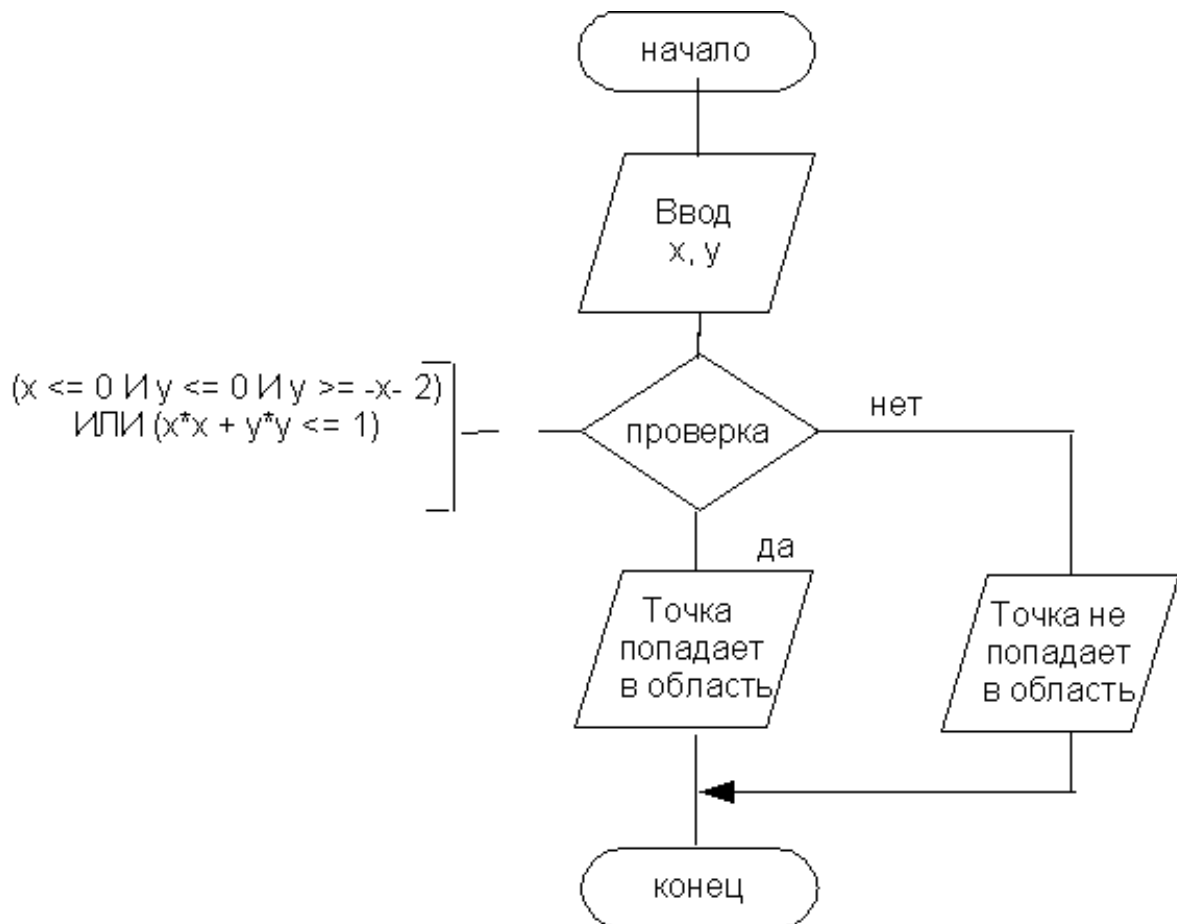


Рисунок 2.2 – Схема программы

Текст программы приведен ниже.

```

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    double x, y;
    /* смена кодировки для вывода русских букв */
    system("chcp 1251");
    /* ввод данных */
    printf("Введите x и y\n");
    scanf("%lf%lf", &x, &y);
    /* Проверка попадания в область */
    if ( x <= 0 && y <= 0 && y >= -x - 2 || x*x + y*y <= 1)
        printf("точка попадает в область\n");
    else printf("точка не попадает в область\n");
    /* ожидание нажатия клавиши */
    system("pause");
    return 0;
}

```

Результаты работы программы:

```

Текущая кодовая страница: 1251
Введите x и y
-1 -1
точка попадает в область
Для продолжения нажмите любую клавишу . . .

```

## 2.5. Содержание отчета и порядок защиты работы

Выполнение и защита лабораторной работы производится каждым студентом индивидуально. Защита результатов лабораторной работы осуществляется при наличии работающей программы и полностью оформленного отчета.

Отчет должен включать в себя следующие разделы

- титульный лист;
- цель работы;
- постановка задачи. Этот раздел должен содержать вариант задания в соответствии с таблицей 2.1;
- схема программы;
- текст программы на языке Си;
- результаты работы программы при хотя бы двух вариантах исходных данных;
- выводы.

Защита работы состоит в следующем:

- представление работающей программы на компьютере;
- предъявление отчета, оформленного в соответствии с указанными требованиями;
- ответы на вопросы преподавателя по теоретической и практической части работы. Примеры возможных вопросов приведены в подразделе 2.6.

## 2.6. Контрольные вопросы

- 1) Как представляются логические значения «истина» и «ложь» в языке Си?
- 2) Поясните термин «логическое выражение».
- 3) Перечислите операции отношения.
- 4) Изобразите таблицу истинности для логических операций языка Си.
- 5) Изобразите таблицу приоритета операций отношения, логических, условной, присваивания.
- 6) Как работает условная операция `?:` в языке Си?
- 7) Каковы синтаксис и алгоритм работы условного оператора `if`?
- 8) Что представляет собой составной оператор в языке Си?
- 9) Каковы синтаксис и алгоритм работы оператора `switch`?
- 10) В каких случаях следует использовать условный оператор `if`? Когда целесообразней использовать оператор выбора `switch`? Приведите примеры.

### **3. ЛАБОРАТОРНАЯ РАБОТА №3. ИССЛЕДОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ НА ЯЗЫКЕ СИ**

#### **3.1. Цель работы**

Целью лабораторной работы является исследование циклических алгоритмов на языке Си и их программирование средствами среды Dev-C++. Для достижения поставленной цели студентам необходимо решить ряд задач: освоить навыки построения итерационных формул для вычисления слагаемых и суммирования функциональных рядов, а также освоить методы программирования итерационных циклических алгоритмов суммирования рядов на языке Си.

#### **3.2. Задание на работу**

1) Работа выполняется в среде Dev-C++. Описание лабораторного стенда приведено в методических указаниях к лабораторной работе № 1.

2) Задания к работе разбиты на две группы и зависят от четности номера варианта (см. табл.3.1). Студенты с четными номерами вариантов выполняют задание из пункта 3.2.1, студенты с нечетными номерами вариантов – из пункта 3.2.2.

##### **3.2.1. Четные варианты заданий**

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда, при фиксированном значении  $x$  и при разных значениях количества слагаемых  $n$ . Число слагаемых  $n$  должно изменяться в цикле от  $n_0$  до  $n_m$  с шагом  $\Delta n$ .

Итак, программа должна:

- 1) ввести исходные данные - значения  $x, n_0, n_m, \Delta n$ ;
- 2) содержать цикл с управляющей переменной  $n$ , изменяющейся от  $n_0$  до  $n_m$  с шагом  $\Delta n$ . В теле цикла должны осуществляться следующие действия:
  - цикл (внутренний), в теле которого для данных  $x$  и  $n$  вычисляется по рекуррентной формуле значение суммы ряда;
  - расчет «точного» значения суммы по формуле согласно варианту задания;
  - вывод на экран значения  $n$ , приближенного и точного значения суммы ряда, ошибки вычисления суммы (разницы между точным и приближенным значением).

##### **3.2.2. Нечетные варианты заданий**

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда, с точностью  $\delta$  на интервале  $x$  от  $x_0$  до  $x_m$  с шагом  $\Delta x$ .

Итак, программа должна:

- 1) ввести исходные данные – значения  $\delta, x_0, x_m, \Delta x$ ;
- 2) содержать цикл с управляющей переменной  $x$ , изменяющейся от  $x_0$  до  $x_m$  с шагом  $\Delta x$ . В теле цикла должны осуществляться следующие действия:
  - цикл, в теле которого для заданных  $x$  и  $\delta$  вычисляется по рекуррентной формуле значение суммы ряда;
  - расчет «точного» значения суммы по формуле согласно варианту задания;
  - вывод на экран значения  $x$ , приближенного и точного значения суммы ряда, ошибки вычисления суммы.

Таблица 3.1 – Варианты заданий

№ варианта	Задания и значения исходных данных		
	Ряд	Функция	Диапазон $x$
1	$(x-1) - \frac{(x-1)^2}{2} + \dots + (-1)^n \frac{(x-1)^{n+1}}{n+1} + \dots$	$\ln x$	0,5; 0,7; ...; 1,5
2	$1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \dots + (-1)^n \frac{x^{2n}}{(2n+1)!} + \dots$	$\frac{\sin(x)}{x}$	-0,5; -0,3; ... 0,5
3	$1 + \frac{\cos(\frac{\pi}{4})}{1!} \cdot x + \dots + \frac{\cos(\frac{\pi}{4})}{n!} \cdot x^n + \dots$	$e^{x \cos(\frac{\pi}{4})} \cos(x \cdot \sin(\frac{\pi}{4}))$	-0,5; -0,3; ... 0,5
4	$1 + 3x^2 + \dots + \frac{2n+1}{n!} x^{2n} + \dots$	$(1 + 2x^2) \cdot e^{x^2}$	0,1; 0,2; ...; 1
5	$\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots + \frac{1}{(2n+1)x^{2n+1}} + \dots$	$\frac{1}{2} \ln \frac{x+1}{x-1}$	1,1; 1,2; ... 2,0
6	$1 + x \ln 3 + \frac{(x \ln 3)^2}{2!} + \dots + \frac{(x \ln 3)^n}{n!} + \dots$	$3^x$	0,1; 0,2; ...; 1
7	$x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2n+1}}{(2n+1)!} + \dots$	$\frac{e^x - e^{-x}}{2}$	1,1; 1,2; ...; 2
8	$1 - 2x + 3x^2 - \dots + (-1)^n (n+1)x^n + \dots$	$\frac{1}{(1+x)^2}$	-0,9; -0,7; ... 0,9
9	$x - \frac{x^2}{2} + \frac{x^3}{3} - \dots + \frac{(-1)^n x^{n+1}}{n+1} - \dots$	$\ln(x+1)$	-0,9; -0,7; ... 0,9
10	$\frac{x-1}{x+1} + \frac{1}{3} \left( \frac{x-1}{x+1} \right)^3 + \dots + \frac{1}{2n+1} \left( \frac{x-1}{x+1} \right)^{2n+1} + \dots$	$\frac{1}{2} \ln x$	0,1; 0,3; ... 2,0
11	$x - \frac{x^3}{3} + \dots + (-1)^n \frac{x^{2n+1}}{2n+1} + \dots$	$\operatorname{arctg}(x)$	-0,9; -0,7; ... 0,9
12	$-x - \frac{x^2}{2} - \frac{x^3}{3} - \dots - \frac{x^n}{n} - \dots$	$\ln(1-x)$	-0,9; -0,7; ... 0,9
13	$x + \frac{x^3}{3} + \frac{x^5}{5} + \dots + \frac{x^{2n+1}}{2n+1} + \dots$	$\frac{1}{2} \ln \frac{1+x}{1-x}$	-0,5; -0,3; ... 0,5
14	$\frac{x^3}{3} - \frac{x^5}{15} + \dots + (-1)^{n+1} \frac{x^{2n+1}}{4n^2 - 1} + \dots$	$\frac{1+x^2}{2} \operatorname{arctg}(x) - \frac{x}{2}$	0,1; 0,2; ...; 1

### 3.3. Краткие теоретические сведения

При математическом описании систем управления, моделей детерминированных и случайных сигналов часто используются функциональные ряды

$$S(x) = \sum_{i=0}^{\infty} V_i(x). \quad (1)$$

При фиксированном значении аргумента  $x$  ряд (1) превращается в числовой ряд. Как известно, числовой ряд называется сходящимся, если предел сумм, составляемых из  $n$  первых его слагаемых, существует при бесконечном увеличении  $n$ . Если при любом значении  $A \leq x \leq B$  ряд (1) сходится, то функциональный ряд называется сходящимся на отрезке  $(A, B)$ . Сумма ряда  $S(x)$  представляет собой функцию аргумента  $x$ . Примеры функциональных рядов и соответствующих им сумм приведены в таблице 2.

Часто аналитическое представление суммы ряда неизвестно, и тогда приходится вычислять ее значение при заданных значениях  $x$  непосредственно вычислением суммы (1) на ЭВМ. Бесконечная сумма заменяется суммой  $n$  слагаемых:

$$S_n(x) = \sum_{i=0}^n V_i(x). \quad (2)$$

Число  $n$  задается или определяется в процессе вычисления, исходя из требований точности суммирования ряда. При этом полная ошибка вычисления  $\varepsilon_n(x)$  (далее для краткости будем называть ее полная ошибка или просто ошибка) представляет собой сумму двух составляющих:

$$\varepsilon_n(x) = \varepsilon_{An}(x) + \varepsilon_{Bn}(x). \quad (3)$$

Первая из них связана с выбором конечного значения  $n$ :

$$\varepsilon_{An}(x) = S(x) - S_n(x). \quad (4)$$

Она называется ошибкой алгоритма. Вторая составляющая  $\varepsilon_{Bn}(x)$  связана с неточностью представления чисел и выполнения операций на ЭВМ. Это ошибка вычислений. С увеличением  $n$  ошибка алгоритма для сходящегося ряда стремится к нулю, а ошибка вычислений в общем случае возрастает. При небольших  $n$  ( $n \leq 100$ ) ошибка вычислений, как правило, во много раз меньше, чем ошибка алгоритма.

Часто встречаются задачи, в которых определить значение  $n$  до составления алгоритма и программы не удастся. Для решения этих задач разрабатываются такие алгоритмы суммирования ряда, в которых количество слагаемых в сумме (2) определяется автоматически в процессе вычислений. Чаще всего применяется следующий способ определения  $n$  – суммирование продолжается до некоторого  $n$  – того слагаемого, при котором в первый раз выполняется условие:

$$|V_n(x)| \leq \delta. \quad (5)$$

Здесь  $\delta$  – некоторое число, которое выбирается, исходя из заданной погрешности вычислений суммы (3).

Следует отметить, что в общем случае число  $\delta$  не определяет значения ошибки  $\varepsilon$  вычисления суммы ряда. Действительно, из выполнения (5), даже если остальные слагаемые монотонно убывают, вовсе не следует, что ошибка суммирования:

$$\left| \sum_{i=n+1}^{\infty} V_i(x) \right| \leq \varepsilon.$$



Поэтому обычно  $\delta$  задается меньшим, чем допустимая погрешность  $\varepsilon$ , или применяется более сложное условие окончания цикла суммирования. Например, суммирование продолжается до слагаемого с номером  $n$ , при котором в первый раз выполняется неравенство

$$\sum_{i=n-k}^n |V_i(X)| \leq \delta, \quad (6)$$

где  $k$ -заданное число. Очевидно, проверять это условие можно лишь при  $n \geq k$ , то есть, когда уже просуммировано не менее  $k$  слагаемых. Алгоритм проверки условия (6) значительно сложнее, чем алгоритм проверки условия (5), и применяется редко.

Вычисление суммы ряда осуществляется в цикле по рекуррентной формуле

$$S_i = S_{i-1} + V_i(x), \quad S_0 = V_0(x). \quad (7)$$

Часто для слагаемых ряда также удастся получить рекуррентные соотношения, выразив последующий член ряда  $V_i(x)$  через предыдущий  $V_{i-1}(x)$ . Это позволяет существенно сократить объем вычислений.

Может оказаться, что удобнее записать рекуррентное соотношение не для всего члена ряда  $V_i(x)$ , а для его части (сомножителя). Ниже в таблице 1 приводятся несколько характерных примеров таких рекуррентных соотношений. Если число слагаемых при суммировании ряда определяется в процессе вычислений, то при программировании пользоваться оператором цикла, при записи которого необходимо указать количество повторений цикла, неудобно. В этом случае удобней организовать возврат к следующему шагу суммирования с помощью условного оператора, который осуществляет проверку условия (5) или (6) окончания цикла.

При программировании удобно применить цикл с предусловием или постусловием. Пример программы для ряда 2 из таблицы 3.2.

Таблица 3.2 – Примеры функциональных рядов

№ п/п	Ряд	Рекуррентная формула для вычисления члена ряда
1	$S(x) = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots$	$v_i = v_{i-1} \cdot \frac{x}{i}, \quad v_0 = 1.$
2	$S(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!} + \dots$	$\begin{cases} l_i = l_{i-1} + 2, & l_0 = 0 \\ v_i = -v_{i-1} \cdot \frac{x^2}{l_i \cdot (l_i - 1)}, & v_0 = 1. \end{cases}$

Для вывода рекуррентного соотношения для члена ряда  $V_i$  или для его части (сомножителя)  $W_i$  рекомендуется воспользоваться следующими приемами:

а) если  $W_i$  представляет собой произведение степеней и факториалов  $i$ :

- записать выражение для  $W_{i-1}$ ;
- записать дробь  $W_i / W_{i-1}$  и получить после сокращения общих сомножителей числителя и знаменателя:

$$\frac{W_i}{W_{i-1}} = k;$$

- записать  $W_i = k \cdot W_{i-1}$  и для  $i = 0$  вычислить начальное условие для  $W_0$ .

б) если  $i$  входит в  $W_i$  в качестве сомножителя:

- записать выражение для  $W_{i-1}$ ;
- записать разность  $W_i - W_{i-1}$  и получить после приведения подобных:

$$W_i - W_{i-1} = d;$$

- записать  $W_i = W_{i-1} + d$  и для  $i = 0$  вычислить начальное условие для  $W_0$ .

(Как правило, такое рекуррентное соотношение позволяет вместо операции умножения использовать в программе менее трудоемкую операцию сложения).

### 3.4. Пример программы

Ниже приведен пример программы, вычисляющей значение функции  $\cos(x)$  с помощью разложения в ряд по формуле

$$S(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^n \frac{x^{2n}}{(2n)!} + \dots$$

для заданного  $x$  с точностью  $\delta$ .

Вывод рекуррентной формулы:

$$v_i = (-1)^i \frac{x^{2i}}{(2i)!};$$

$$v_{i-1} = (-1)^{i-1} \frac{x^{2(i-1)}}{(2(i-1))!};$$

$$\frac{v_i}{v_{i-1}} = \frac{(-1)^i x^{2i} \cdot (2(i-1))!}{(2i)! (-1)^{i-1} x^{2(i-1)}} = - \frac{x^2 \cdot 1 \cdot 2 \cdot 3 \cdot \dots \cdot (2i-2)}{1 \cdot 2 \cdot 3 \cdot \dots \cdot (2i-2) \cdot (2i-1) \cdot 2i} = - \frac{x^2}{(2i-1) \cdot 2i};$$

Обозначим  $l_i = 2i$ , тогда  $l_{i-1} = 2(i-1)$ , следовательно,  $l_i - l_{i-1} = 2i - 2(i-1) = 2$ , откуда

$$l_i = l_{i-1} + 2, l_0 = 0.$$

Окончательно получаем:

$$v_i = -v_{i-1} \cdot \frac{x^2}{l_i \cdot (l_i - 1)}; v_0 = 1.$$

Текст программы имеет следующий вид.

```
#include <stdio.h>
#include <math.h>
int main()
{
    double x, s, delta, v, k;
    int l, i //счетчик, вспомогательная переменная
    printf("x=");
    scanf("%lf", &x);
    printf("delta=");
    scanf("%lf", &delta);
    v = 1.0; s = v; l = 0; k = x*x;
    for (i = 1; fabs(v) > delta; i++)
    {
        l += 2; v *= -k/l/(l-1); s += v;
    }
    printf("s= %6lf, cos x = %6lf, n = %4d\n", s, cos(x), i);
    fflush(stdin);
    getchar(); return 0; }
```

### 3.5. Содержание отчета и порядок защиты работы

Выполнение и защита лабораторной работы производится каждым студентом индивидуально. Защита результатов лабораторной работы осуществляется при наличии работающей программы и полностью оформленного отчета.

Отчет должен включать в себя следующие разделы

- титульный лист;
- цель работы,
- постановка задачи. Этот раздел должен содержать вариант задания в соответствии с таблицей 3.1;
- схема программы. Этот раздел должен содержать схему программы, а также схемы разработанных функций.
- текст программы на языке Си;
- результаты работы программы;
- выводы.

Защита работы состоит в следующем:

- представление работающей программы на компьютере;
- предъявление отчета, оформленного в соответствии с указанными требованиями;
- ответы на вопросы преподавателя по теоретической и практической части работы. Примеры возможных вопросов приведены в подразделе 3.6.

### 3.6. Контрольные вопросы

- 1) Опишите синтаксис и алгоритм работы оператора цикла `while` в языке Си.
- 2) Опишите синтаксис и алгоритм оператора цикла `do...while` в языке Си.
- 3) Опишите синтаксис и алгоритм работы оператора цикла `for` в языке Си.
- 4) От каких исходных данных зависит время выполнения составленных в лабораторной работе программ, и каким образом?
- 5) Какие циклы называются вложенными?
- 6) Какие блоки в двойном циклическом алгоритме выполняются наибольшее число раз при любых исходных данных?
- 7) Какие составляющие влияют на величину полной ошибки суммирования ряда?
- 8) Поясните способы выбора количества слагаемых суммы ряда?

## 4. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Керниган Б. Язык программирования Си /Б. Керниган, Д. Ритчи. – СПб.: "Невский Диалект", 2001. – 352 с.
2. Павловская Т.А. C/C++. Программирование на языке высокого уровня /Т.А. Павловская. – СПб.:Питер, 2003. – 461 с.
3. Павловская Т.А. C/C++. Структурное программирование: Практикум /Т.А. Павловская, Ю.А. Щупак. – СПб.:Питер, 2005. – 239 с.
4. Шпак З.Я. Програмування мовою С. – Львів: Оріяна-Нова, 2006. – 432 с.
5. Дейтел П. Как программировать на С /П. Дейтел, Х. Дейтел. – М.: Изд-во «Бином», 2006.– 912 с.

## **ПРИЛОЖЕНИЕ А.**

### **ПЕРЕЧЕНЬ ТЕМ БЛОКА №4**

(справочное)

- 1) История языка Си.
- 2) Стандартная библиотека языка Си.
- 3) Этапы обработки программы на языке Си.
- 4) Структура программы на Си. Пример простой программы на Си: сложение двух чисел.
- 5) Основы синтаксиса языка Си. Алфавит и основные лексемы языка Си.
- 6) Базовые типы данных языка Си.
- 7) Основы синтаксиса языка Си. Переменные и константы.
- 8) Приведение типов в выражениях на языке Си.
- 9) Условный оператор в языке Си. Примеры.
- 10) Логические операции. Операции сравнения. Условная операция в языке Си. Примеры.
- 11) Управляющие структуры. Оператор безусловного перехода `goto`. Пустой оператор. Составной оператор.
- 12) Оператор `switch`. Пример.
- 13) Операции присваивания в языке Си. Примеры.
- 14) Операции инкремента и декремента. Примеры.
- 15) Оператор цикла `while`. Пример.
- 16) Оператор цикла `do/while`. Пример.
- 17) Оператор цикла `for`. Пример.
- 18) Операторы `break` и `continue`. Примеры.