МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методические рекомендации к лабораторным работам для студентов специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)» дневной формы обучения

Часть 1

Могилев 2019

УДК 004.4
БКК 32.973
О 75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «12» июня 2019 г., протокол № 12

Составители: ст. преподаватель А. И. Кашпар;
ст. преподаватель О. В. Сергиенко;
доц. Н. Н. Горбатенко

Рецензент Ю. С. Романович

Методические рекомендации к лабораторным работам предназначены для студентов специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)» дневной формы обучения.

Учебно-методическое издание

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Часть 1

Ответственный за выпуск	А. И. Якимов
Технический редактор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 31 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, Могилёв,

Содержание

Введение

Дисциплина «Основы алгоритмизации и программирования» формирует у студентов основополагающие знания, умения и навыки в области алгоритмизации вычислительных процессов, принципов разработки, тестирования и анализа программного кода на языке высокого уровня C++, необходимые для успешного изучения дисциплин профессионального цикла и дальнейшей профессиональной деятельности.

С целью закрепления теоретического материала и приобретения студентами практических навыков программирования рабочей программой дисциплины предусмотрено проведение лабораторных работ.

Методические рекомендации содержат цикл лабораторных работ в объёме 34 часов аудиторных занятий, предназначенный для выполнения в осеннем семестре студентами первого курса дневной формы обучения специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)». Для студентов заочной сокращенной формы обучения в осеннем семестре обязательны к выполнению лабораторные работы № 5, 7 и 10.

Отчет по лабораторным работам оформляется индивидуально каждым студентом и выполняется на листах формата А4. В состав отчета входят: титульный лист; цель работы; текст индивидуального задания; выполнение индивидуального задания (код программы и блок-схема алгоритма).

1 Лабораторная работа № 1. Алгоритмы

Цель работы:
– программирование базовых конструкций алгоритмов;
– получение практических навыков по работе с алгоритмами.

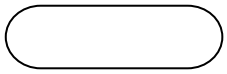
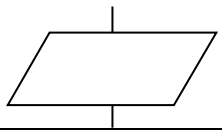
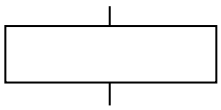
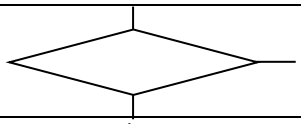
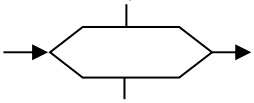
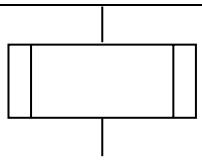
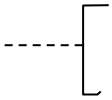
Постановка задачи: составить блок-схемы для четырех заданий.

1.1 Алгоритмизация

Алгоритм – это строго определенная последовательность действий, необходимых для решения данной задачи. Общепринятыми способами записи являются графическая запись с помощью блок-схем и символьная запись с помощью какого-либо алгоритмического языка.

Описание алгоритма с помощью блок схем осуществляется рисованием последовательности геометрических фигур, каждая из которых подразумевает выполнение определенного действия алгоритма. Внешний вид основных блоков, применяемых при написании блок схем, приведен в таблице 1.1.

Таблица 1.1 – Графические изображения элементов схем алгоритмов

Символ	Значение	Применение
	Завершение	Начало, конец обработки данных или выполнения программы
	Данные	Обозначает ввод, вывод данных
	Процесс	Обработка данных любого вида (выполнение операции или группы операций)
	Решение	Выбор направления выполнения программы в зависимости от некоторых переменных условий
	Подготовка	Описывается подготовка данных для выполнения повторяющихся действий
	Типовой процесс	Одна или несколько операций, которые определены в другой программе, модуле
	Соединитель (узел)	Используется при разрыве линий схемы алгоритма
	Комментарий	Используется для пояснений

Основные алгоритмические конструкции

Линейный алгоритм – алгоритм или фрагмент алгоритма, в котором порядок исполнения инструкций соответствует порядку их записи.

Инструкции линейного алгоритма выполняются последовательно одна за другой в порядке их записи. Блок-схему линейного алгоритма обычно представляют в виде блок-символов, соединенных последовательно. В каждый блок-символ входит не более одной линии потока информации. Из каждого блок-символа выходит не более одной линии потока информации. Обычно блок-схему размещают таким образом, что блок-символы размещаются один под другим, и нет необходимости обозначать линии потока информации стрелками.

Разветвляющийся алгоритм – алгоритм или фрагмент алгоритма, в котором порядок исполнения инструкций не определен изначально, имеет не менее двух вероятных направлений, каждое из которых соответствует одному из возможных исходов проверки логического условия.

Своим ветвлением разветвляющийся алгоритм обязан логическому условию, проверка которого на истинность обязательно приводит к возможности реализации одного из двух взаимоисключающих исходов. Линии потока информации, выходящие из блока «Решение», могут быть отмечены парами противоположных по смыслу знаков, представленных в таблице 1.2.

Таблица 1.2 – Знаки и символы, используемые в алгоритмах

Обозначение	Назначение
«да», «нет» «+», «–» «1», «0»	Условие истинно: «да», «+», «1» Условие ложно: «нет», «–», «0 »

Стандартное ветвление имеет два вероятных направления (рисунок 1.1).

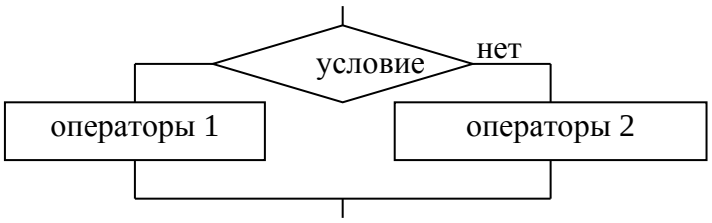


Рисунок 1.1 – Стандартное ветвление

В то же время, комбинируя проверку нескольких логических условий, можно получить многоуровневое ветвление с тремя и более вероятными направлениями.

Циклический алгоритм – фрагмент алгоритма с ветвлением, в котором инструкция или группа инструкций исполняются более одного раза, т. е. повторяются.

Тело цикла – группа повторяющихся инструкций.

Основой циклического алгоритма является повторение инструкции или группы инструкций – тела цикла. Начало и завершение повторения тела цикла определяется итогом проверки логического условия.

Циклический алгоритм имеет две базовые структуры – **цикл с постусловием** и **цикл с предусловием**.

Циклический алгоритм с постусловием (рисунок 1.2) характеризуется тем, что проверка условия расположена после тела цикла (лат. post – после).

Циклический алгоритм с предусловием характеризуется тем, что проверка условия расположена перед телом цикла (рисунок 1.3).

Циклический алгоритм с параметром применяется, когда известно точное число повторений цикла (рисунок 1.4).

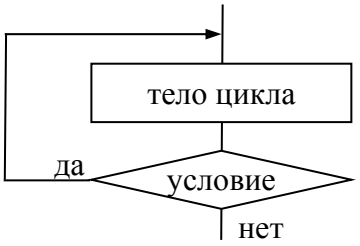


Рисунок 1.2 – Циклический алгоритм с постусловием

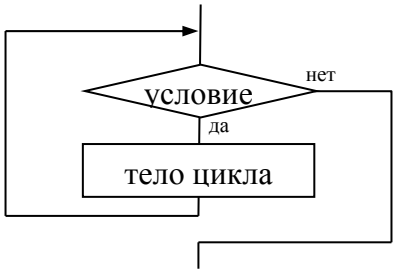


Рисунок 1.3 – Циклический алгоритм предусловием

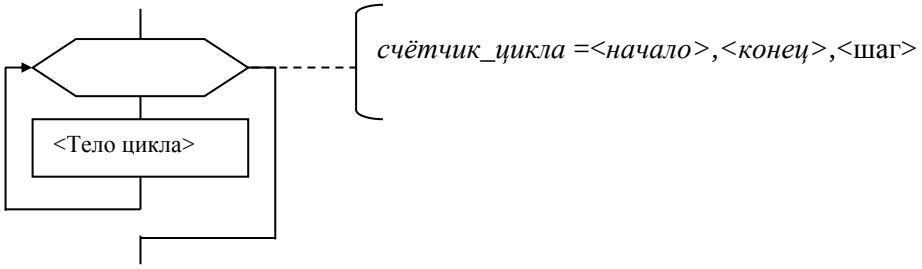


Рисунок 1.4 – Циклический алгоритм с параметром

1.2 Примеры алгоритма

Пример 1 – Образец составления алгоритма решения квадратного уравнения $ax^2 + bx + c = 0$.

Перед составлением алгоритма надо четко определить, что в него требуется ввести и что должно получиться в результате (рисунок 1.5).

В данном случае:

– в качестве исходных данных выступают три вещественных числа: a, b и c (коэффициенты квадратного уравнения);

– в качестве результата — корни квадратного уравнения, вещественные числа x1 и x2.

Пример 2 – Вычислить n-й член последовательности, заданной формулой $a_n = a_{n-1} + a_{n-2}$, если $a_1 = 1$, $a_2 = 1$. Пример составления разветвляющего алгоритма представлен на рисунке 1.6.

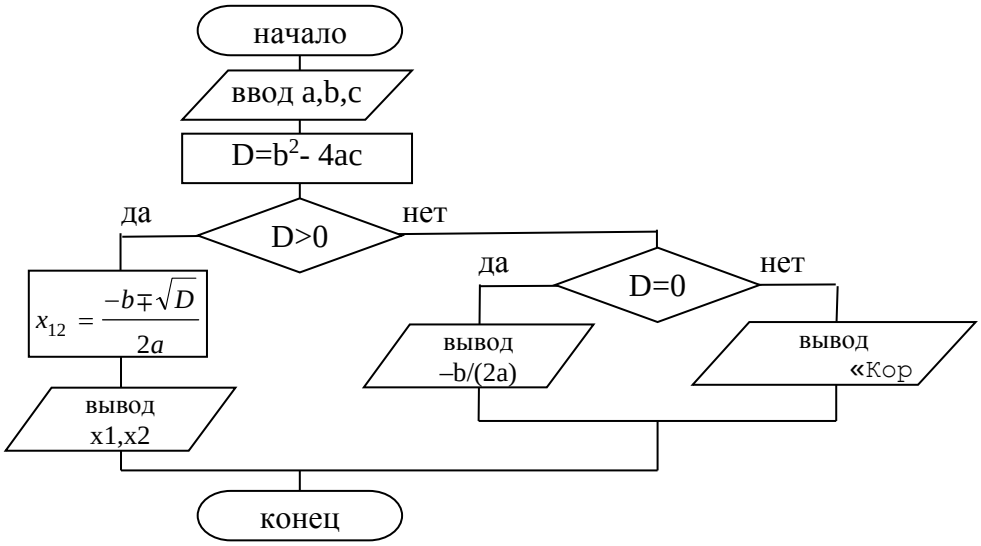


Рисунок 1.5 – Пример составления разветвляющего алгоритма

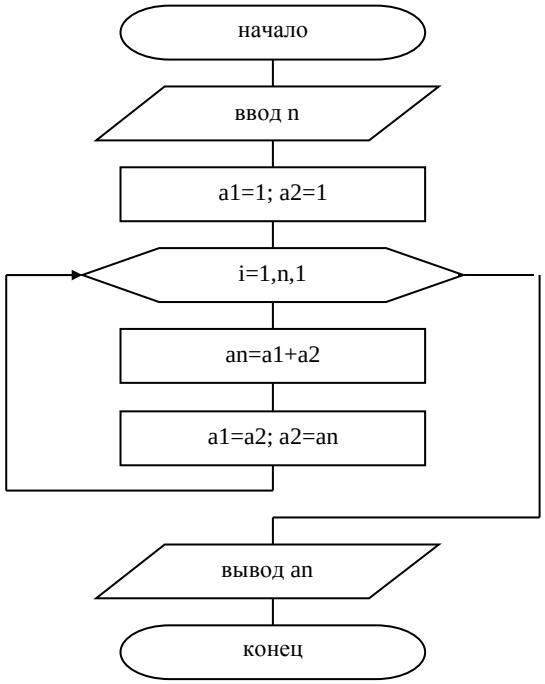


Рисунок 1.6 – Пример составления циклического алгоритма

Задание 1

Составить блок-схему алгоритма решения задачи с условным переходом. Необходимо рассчитать значение искомой переменной по одному из двух альтернативных выражений в зависимости от значения переменной условия, значение которого необходимо предварительно вычислить согласно заданию. Значения переменной условия и переменной результата должны выводиться на экран.

Исходные данные к заданию находятся в таблице 1.3.

Таблица 1.3 – Индивидуальные задания

Номер задания	Данные 1	Данные 2	Переменная условия	Условие выполняется	Условие не выполняется
1	$A_1=2.35$	$A_2=-5.89$	$B=(A_1^2-A_2)>0$	$C=3.1B-A_1A_2^2$	$C=A_1^2A_2-3.1B$
2	$X_1=-8.44$	$X_2=1.73$	$Y=(4X_1-X_2)<0$	$Z=5Y^2-X_2+2X_1$	$Z=X_1+X_2-5Y^2$
3	$E_1=7.54$	$E_2=-3.62$	$P=(3E_2-E_2)>0$	$Y=4F/(E_1-E_2)$	$Y=4F/(E_2-E_1)$
4	$B_1=-6.71$	$B_2=4.57$	$E=(E_1-E_2)<0$	$X=2E-B_1+B_2^2$	$X=2E-B_2+B_1^2$
5	$C_1=3.26$	$C_2=-5.41$	$X=(C_3C_2)>0$	$E=7.5X-C_1/C_2$	$E=C_1/C_2-7.5X$
6	$D_1=-9.08$	$D_2=6.35$	$I=(D_1-D_2^2)<0$	$B=2.1D_1-3.4I$	$B=1.2D_2-3.4I$
7	$F_1=5.12$	$F_2=-2.06$	$A=(7F_1^2-F_2)>0$	$D=3.1A-7F_1F_2$	$D=7F_1F_2-3.1A$
8	$H_1=-0.97$	$H_2=7.94$	$D=(H_1/H_2^2)<0$	$F=8D-7.5H_1H_2$	$F=4.5H_1H_2-6D$
9	$I_1=1.63$	$I_2=-9.18$	$H=(3I_1^2-I_2^2)>0$	$A=H/(3.8I_1-I_2)$	$A=H/(I_2-3.8I_1)$
10	$J_1=-4.32$	$J_2=0.79$	$C=(J_1J_2-9)<0$	$I=3C-4.1(J_1J_2)^2$	$I=4(J_1J_2)^2-3C$
11	$K_1=7.45$	$K_7=-3.82$	$J=(K_1-5K_2^2)>0$	$H=2.5J-K_1K_2$	$H=K_1K_2-2.5J$
12	$L_1=-2.71$	$L_2=5.63$	$K=(L_1^2/L_2)<0$	$J=9K-3L_1+L_2$	$J=3L_1-L_2-9K$

Задание 2

1 Даны три целых числа. Найти количество положительных чисел в исходном наборе.

2 Даны три целых числа. Найти количество положительных и количество отрицательных чисел в исходном наборе.

3 Даны две переменные вещественного типа: A, B. Перераспределить значения данных переменных так, чтобы в A оказалось меньшее из значений, а в B – большее. Вывести новые значения переменных A и B.

4 Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной сумму этих значений, а если равны, то присвоить переменным нулевые значения. Вывести новые значения переменных A и B.

5 Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной большее из этих значений, а если равны, то присвоить переменным нулевые значения. Вывести новые значения переменных A и B.

6 Даны три числа. Найти наименьшее из них.

7 Даны три числа. Найти среднее из них (то есть число, расположенное между наименьшим и наибольшим).

8 Даны три числа. Вывести вначале наименьшее, а затем наибольшее из данных чисел.

9 Даны три числа. Найти сумму двух наибольших из них.

10 Даны три переменные вещественного типа: A, B, C. Если их значения упорядочены по возрастанию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных A, B, C.

11 Даны три переменные вещественного типа: A, B, C. Если их значения упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных A, B, C.

12 Даны три целых числа, одно из которых отлично от двух других, равных между собой. Определить порядковый номер числа, отличного от остальных.

Задание 3

Составить блок-схему алгоритма решения циклической задачи.

1 Дано целое число N > 0. Найти сумму 1 + 1/2 + 1/3 + ... + 1/N.

2 Дано целое число N > 0. Найти сумму $N^2 + (N + 1)^2 + (N + 2)^2 + \ldots + (2N)^2$.

3 Дано целое число N > 0. Найти произведение $1 \cdot 2 \cdot 3 \cdot \ldots \cdot N$.

4 Дано целое число N > 0. Найти значение выражения 1,1 – 1,2 + 1,3 – ... (N слагаемых, знаки чередуются).

5 Даны вещественное число A и целое число N > 0. Найти A в степени N (числа A перемножаются N раз).

6 Даны вещественное число A и целое число N > 0. Найти сумму $1 + A + A^2 + A^3 + \ldots + A^N$.

7 Даны вещественное число A и целое число N > 0. Найти значение выражения $1-A+A^2-A^3 + \ldots + (-1)^N A^N$.

8 Даны целое число N>0. Найти произведение N! = $1 \cdot 2 \cdot \ldots \cdot N$ (N-факториал). Чтобы избежать целочисленного переполнения, вычислять это произведение с помощью вещественной переменной и вывести его как вещественное число.

9 Дано целое число N > 0. Найти сумму 1! +2! + 3! + ...+N! (выражение N! – N-факториал обозначает произведение всех целых чисел от 1 до N). Чтобы избежать целочисленного переполнения, проводить вычисления с помощью вещественных переменных и вывести результат как вещественное число.

10 Дано целое число N > 0. Найти сумму 1 + 1/(1!) + 1/(2!) + 1/(3!) + ... + + 1/(N!) (выражение N! – N-факториал обозначает произведение всех целых чисел от 1 до N). Полученное число является приближенным значением константы e.

11 Дано вещественное число X и целое число N > 0. Найти значение выражения $1 + X + X^2/(2!) + \ldots + X^N/(N!)$ ($N! = 1 \cdot 2 \cdot \ldots \cdot N$). Полученное число является приближенным значением функции exp в точке X.

12 Даны вещественное число X ($|X|<1$) и целое число N > 0. Найти значение выражения $X – X^2/2 + X^3/3 – \ldots + (-1)^{N-1} \cdot X^N/N$. Полученное число является приближенным значением функции ln в точке 1+X.

2 Лабораторная работа № 2. Работа с главным меню системы Visual C++. Структура программы. Программирование линейных алгоритмов

Цель работы:

– изучение интегрированной среды;
– освоение простейшей структуры программы на языке C++;
– получение навыков в организации ввода-вывода на языке C++;
– изучение базовых типов данных языка C++.

Постановка задачи: написать программу для расчета по двум формулам.

2.1 Общие сведения

После запуска интегрированной среды Microsoft Visual C++ 6.0 необходимо создать проект. Для этого выполняем следующую последовательность команд:

1) File > New...;
2) в открывшемся окне:
   а) выберите тип Win32 Console Application;
   б) введите имя проекта в текстовом поле Project Name;
   в) введите (выберете с помощью кнопки …) имя каталога размещения файлов проекта в текстовом поле Location, например G:/ASOIZ/;
   г) щелкните левой кнопкой мыши на кнопке ОК;
   д) открывается диалоговое окно Win32 Console Application – Stepl of 1 и в нем выберите тип An empty project и щелкните на кнопке Finish.
3) появится окно New Project, в котором щелкните на кнопке ОК.

Далее создадим файл:

1) File > New.... В результате откроется диалоговое окно New;
2) на вкладке Files:
   а) выберите тип файла (в данном случае: C++ Source File);
   б) в текстовом поле File Name введите нужное имя файла;
   в) флажок Add to project должен быть включен и нажмите кнопку ОК.

В открывшемся редакторе можно создавать программу, но прежде необходимо ознакомиться со структурой простейшей программы на языке C++.

Любая программа на C++ состоит из функций, одна из которых должна иметь имя main, указывающее, что именно с нее начинается выполнение программы. После круглых скобок в фигурных скобках { } записывается тело функции, т. е. те операторы, которые требуется выполнить. Любая заготовка при написании программы имеет вид:

```
#include <…>
int main()
{
 объявление переменных;
 ввод исходных данных;
 расчет результата;
 вывод результата;
  return 0;
}
```

В начале программы записываются директивы препроцессора, по которой к исходному тексту программы подключается заголовочный файл для использования тех или иных функций (например, директива **#include <iostream.h>** содержит описания операторов ввода-вывода cin и cout, для использования в программе математических функций необходимо в начале программы поместить директиву **#include <math.h>**);

Для хранения данных в программе надо выделить достаточно места в оперативной памяти. Для этого необходимо объявить переменные. Синтаксис объявления переменных:

тип имя_переменной;

Имена переменным дает программист исходя из их назначения. Имя может состоять только из латинских букв, цифр и знака подчеркивания и должно начинаться не с цифры.

Тип данных определяет:

– множество значений, которые могут принимать величины этого типа;

– внутреннее представление данных в памяти компьютера;

– операции и функции, которые можно применять к величинам этого типа.

Основные типы:

int (short, unsigned) – целочисленные;

float (double, long double) – вещественные;

char – символьный;

bool – логический.

Для ввода-вывода информации используются операторы cin и cout. Рассмотрим программу:

```
#include <iostream.h>
int main()
{
  int i;
  cout<<"Введите целое число:"<<endl;
  cin>>i;
  cout<<"Вы ввели число "<<i<<", спасибо!"<<endl;
  return 0;}
```

Первая строка этой программы – директива препроцессора, по которой в текст программы вставляется заголовочный файл < iostream.h>, содержащий описание использованных в программе операторов ввода/вывода.

Третья строка — описание переменной целого типа с именем i. Оператор cout в четвертой строке выводит приглашение «Введите целое число» и переходит на новую строку (endl). Оператор cin заносит введенное с клавиатуры целое число в переменную i, а следующий оператор cout выводит на экран указанные в нем строки (указаны в кавычках), а между ними – значение переменной i (имя переменной без кавычек).

Прокомпилируем программу. Для этого нажимаем кнопку  на панели инструментов либо комбинацию клавиш Ctrl+F7. В окне вывода (внизу экрана) должно вывестись сообщение 0 error(s), 0 warning(s) (0 ошибок, 0 предупреждений). Двойной щелчок по тексту ошибки указывает месторасположение ошибки в программе. Если есть ошибки, сверьте с оригиналом.

Для запуска программы нажимаем кнопку на панели инструментов либо комбинацию клавиш Ctrl+F5.

При запуске программы вместо русских символов видим совсем другие изображения, что вызвано различными стандартами кодировки символов кириллицы в операционных системах MS DOS и Windows. Для исправления можно использовать латинские символы либо добавить в программу функцию CharToOem (дополнения для наглядности выделены жирным шрифтом).

```
#include <iostream.h>
#include <windows.h> //для использования функции CharToOem
char  buf[256];
```

```
char* RUS(const char* text)
{   CharToOem(text, buf);
  return buf;}
int main()
{
  int i;
  cout<<RUS("Введите целое число:")<<endl;
  cin>>i;
  cout<<RUS("Вы ввели число ")<<i;
  cout<<RUS(", спасибо!")<<endl;
  return 0; }
```

Если в программе все операторы выполняются последовательно, один за другим, такая программа называется *линейной*.

Одна из основных операций языка С++ – это операция присваивания. В качестве ее левого операнда может использоваться только модифицируемое именующее выражение, т. е. ссылка на некоторую именованную область памяти, значение которой доступно изменениям.

Существуют одна простая операция присваивания и ряд составных.

Простое присваивание =

A = B*2; // присвоить переменной A результат вычисления выражения B*2.

Все составные операции присваивания представляют собой сокращенную запись простого присваивания:

– для случая, который может быть описан как X = XÅY;

– то же, в записи составного присваивания X Å=Y.

В качестве символа Å могут быть использованы знаки операций: +; -; *; /; % и др.

Например, f += 5; то же самое, что и f = f + 5.

В задачах вычислительного типа над переменными и константами могут выполняться арифметические операции и функции.

В таблице 2.1 представлены основные арифметические операции.

Таблица 2.1 – Основные арифметические операции

Выражение	Операция		Пример		
			A	B	Результат
a +b	Сложение		5	2,75	7,75
a – b	Вычитание		5	2,75	2,15
a*b	Умножение		2	6	12
a/b	Деление	a и b – целые	7	2	3
		в остальных случаях	7	2	3.5
a % b	Остаток от деления по модулю		7	2	1

Стандартные математические функции представлены в таблице 2.2 (для использования математических функций необходимо подключить к программе заголовочный файл <math. h>).

Таблица 2.2 – Стандартные математические функции

Обращение	Функция	Обращение	Функция
abs(n)	Возвращает модуль целого числа	atan (x)	Арктангенс(результат в радианах)
fabs (x)	Возвращает модуль вещественного числа	log(x)	Натуральный логарифм от x
sin(x)	Синус угла x (x – в радианах)	log10(x)	Логарифм по основанию 10 от x
cos(x)	Косинус угла x (x – в радианах)	pow(x,y)	Возводит число x в степень y
tan(x)	Тангенс угла x (x – в радианах)	exp(x)	Степень числа e (экспонента)
asin(x)	Арксинус(результат в радианах)	sqrt(x)	Квадратный корень из числа
acos(x)	Арккосинус(результат в радианах)		

Пример **–** Вычислить значение функции

$$f(x, y) = \frac{|x| + \sin^3(y + 5)}{x + \frac{1}{3}}.$$

Перед написанием любой программы надо четко определить, что в нее требуется ввести и что должно получиться в результате.

В данном случае:

– в качестве исходных данных выступают два вещественных числа: x и y;

– в качестве результата – значение функции, вещественное число f.

Текст программы:

```
//директивы препроцессора
#include <iostream.h>    //для использования cin, cout
#include <math.h>        // для использования fabs(), pow(), sin()
#include <windows.h>   //для использования CharToOem()
//функция для вывода кириллицы в консоли
char  buf[256];
char* RUS(const char* text)
{
  CharToOem(text, buf);
  return buf;
}
int main()
{
  float x,y,f;   //объявление переменных вещественного типа
  cout<<RUS("Введите x и y:")<<endl; //вывод приглашения к вводу
  cin>>x>>y;  //ввод исходных данных
  f=(fabs(x)+pow(sin(y+5),3))/(x+1./3);  //вычисление результата
  cout<<RUS("Результат: f(x,y)= ")<<f<<endl;   //вывод результата
  return 0;   }
```

Задание 1

Написать программу для расчета по двум формулам. Предварительно подготовьте тестовые примеры по второй формуле с помощью калькулятора (результат вычисления по первой формуле должен совпадать со второй).

1) $z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha)$;

$z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right)$;

2) $z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$;

$z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right)$;

3) $z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$;

$z_2 = 2\sin\alpha$;

4) $z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \frac{\cos 2\alpha}{1 + \cos 2\alpha}$;

$z_2 = \operatorname{ctg}\left(\frac{3}{2}\pi - \alpha\right)$;

5) $z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha$;

$z_2 = \cos^2\alpha + \cos^4\alpha$;

6) $z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$;

$z_2 = 4\cos\frac{\alpha}{2} \cdot \cos\frac{5}{2}\alpha \cdot \cos 4\alpha$;

7) $z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$;

$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$;

8) $z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1$;

$z_2 = \sin(y + x) \cdot \sin(y - x)$;

9) $z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$;

$z_2 = -4\sin^2\frac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta)$;

10) $z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$;

$z_2 = \operatorname{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$;

11) $z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}$; $z_2 = \frac{1 - \operatorname{tg}\alpha}{1 + \operatorname{tg}\alpha}$;

12) $z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$;

$z_2 = \operatorname{tg} 3\alpha$.

Задание 2

1 Ввести с клавиатуры длины катетов a и b прямоугольного треугольника. Найти его периметр и площадь.

2 Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

3 Найти длину окружности и площадь круга заданного радиуса R.

4 Найти площадь кольца, внутренний радиус которого равен R1, а внешний радиус равен R2 (R1 < R2). В качестве значения Pi использовать 3.14.

5 Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей.

6 Дана длина окружности. Найти площадь круга, ограниченного этой окружностью. В качестве значения Pi использовать 3.14.

7 Найти расстояние между двумя точками с заданными координатами (x1, y1) и (x2, y2).

8 Ввести с клавиатуры координаты трех вершин треугольника (x1, y1), (x2, y2), (x3, y3). Найти его периметр и площадь.

9 Найти действительные корни квадратного уравнения $A \cdot x^2 + B \cdot x + C = 0$, заданного своими коэффициентами A, B, C (коэффициент A не равен 0), если известно, что уравнение имеет ровно два корня.

10 Дано целое четырехзначное число. Используя операции деления / и нахождения остатка от деления %, найти сумму и произведение его цифр.

11 Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч (U < V). Время движения лодки по озеру T1 ч, а по реке (против течения) –

T2 ч. Определить путь S, пройденный лодкой.

12 Скорость первого автомобиля V1 км/ч, второго – V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга.

3 Лабораторная работа № 3. Программирование разветвляющихся алгоритмов

Цель работы:

– закрепление структуры программы на языке C++;
– повторить базовые типы данных и математические функции;
– получение навыков в программирование разветвляющихся алгоритмов.

Постановка задачи: написать программу для расчета по нескольким формулам в зависимости от заданных условий.

3.1 Общие сведения

Большинство операторов управления программой в любых языках программирования, включая C++, основываются на проверке условий, определяющих, какого рода действие необходимо выполнить. В результате проверки условий можно получить истину или ложь. В противоположность другим языкам, где вводится специальный тип для хранения истины и лжи, в C++ истине соответствует любое ненулевое значение, включая отрицательные числа, лжи соответствует ноль.

Стандартная форма записи оператора **if** следующая:

if (*выражение*)
оператор;
else
оператор;

где *оператор* может быть простым или составным (надо помнить, что в C++ *составной оператор* – это группа операторов, заключенных в фигурные скобки). Оператор **else** не обязателен.

В условии могут использоваться следующие операции.

Операции отношений (сравнения):

<	меньше, чем	>=	больше или равно
>	больше, чем	==	равно
<=	меньше или равно	!=	не равно

Операнды в этих операциях должны быть арифметического типа или указателями. Результат операции логический (целочисленный): 0 (ложь) или 1 (истина).

Логические операции:

&& конъюнкция (И) арифметических операндов или отношений. Результат истина (1), если два операнда имеют значение истина (не 0);

|| дизъюнкция (ИЛИ) арифметических операндов или отношений. Результат истина (1), если хотя бы один из операндов истина (не 0).

Примеры отношений и логических операций:

4 < 9 (≡ true)
3 = = 5 (≡ false)
3 != 5 || 3==5 (≡ true)
(3+4>5) && (3+5>4) && (4+5>3) (≡ true)

Стандартная форма оператора **if** с составными операторами следующая:

```
if (выражение)
{
    последовательность операторов1
}
else
{
    последовательность операторов2
}
```

Если выражение истинно (любое значение, кроме 0), выполняется блок операторов, следующий за **if**; иначе выполняется блок операторов, следующий за **else**. Всегда выполняется код, ассоциированный или с **if**, или с **else**, но никогда не выполняются два кода одновременно.

На схеме алгоритма оператор **if** (рисунок 3.1).

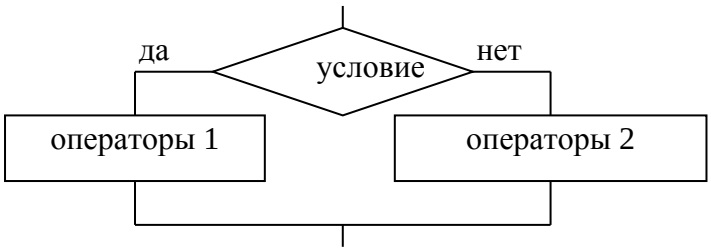


Рисунок 3.1 – Изображение оператора if на схеме алгоритма

Вложенные операторы if.

Типичной программистской конструкцией является *вложенные if*. Данная конструкция выглядит следующим образом:

```
if (выражение)
  оператор;
else if (выражение)
  оператор;
…
else
  оператор;
```

Условия вычисляются сверху вниз. Когда обнаруживается истинное условие, то выполняется оператор, связанный с этим условием, а остальная часть конструкции игнорируется. Если не найдено ни одного истинного условия, выполняется оператор, соответствующий последнему **else**. Последний оператор **else** часто играет роль *оператора*, *выполняемого по умолчанию*, то есть если все условия ложны, то выполняется оператор, соответствующий последнему **else**. Если последний оператор **else** отсутствует, то не выполняется никаких

действий в случае ложности всех условий.

При написании программы с использованием условного оператора обратите внимание на следующие моменты:

– условие должно быть в круглых скобках;

– после условия точка с запятой не ставится (если это не пустой оператор);

– если к if или к else относится более одного оператора, то они объединяются в операторные скобки { };

– в условии проверки на равенство должна использоваться операция сравнения (==);

– условие принадлежности диапазону a < x < b записывается в виде

if(x > a&&x < b) …

Пример **–** Найти корни квадратного уравнения вида $ax^2 + bx + c = 0$. Если корней нет, вывести соответствующее сообщение.

В данной задаче:

– в качестве исходных данных выступают три вещественных числа a, b и c (коэффициенты квадратного уравнения);

– в качестве результата — корни квадратного уравнения, вещественные числа x1 и x2.

Текст программы:

```
//директивы препроцессора
#include <iostream.h>    //для использования cin, cout
#include <math.h>        // для использования sqrt(), pow()
#include <windows.h>   //для использования CharToOem()
//функция для вывода кириллицы в консоли
char  buf[256];
char* RUS(const char* text)
{           CharToOem(text, buf);    return buf;  }
int main()
{
  float a,b,c,x1,x2,D;   //объявление переменных вещественного типа
  cout<<RUS("Введите a,b и c:")<<endl; //вывод приглашения к вводу
  cin>>a>>b>>c;  //ввод исходных данных
  D=pow(b,2)+4*a*c;  //вычисление дискриминанта
  if(D>0)        {
  x1=(-b-sqrt(D))/(2*a);
  x2=(-b+sqrt(D))/(2*a);
  //вывод результата
  cout<<RUS("Уравнение имеет два корня x1=")<<x1<<", x2= "<<x2<<endl;
}
else if(D==0)
  cout<<RUS("Уравнение имеет корень x1= ")<<-b/(2*a)<<endl; //вывод результата
else
  cout<<RUS("Уравнение не имеет корней")<<endl; //вывод результата
return 0;
}
```

Задание 1

1 Написать программу для вычисления значения функции.

$$z=\begin{cases}\sin\dfrac{x+y}{y}, & \text{при } y\neq 0 \quad \text{и} \quad |x|>|y|;\\ \arccos\dfrac{x}{y}, & \text{при } y\neq 0 \quad \text{и} \quad |x|\le|y|;\\ \pi, & \text{в остальных случаях.}\end{cases}$$

2 Написать программу для вычисления значения функции.

$$y=\begin{cases}\cos x, & \text{при } x\le 0;\\ \arcsin x, & \text{при } 0<x\le\dfrac{\pi}{2};\\ \log_4 x, & \text{при } \dfrac{\pi}{2}<x\le 64;\\ \dfrac{1}{x^2}, & \text{в остальных случаях.}\end{cases}$$

3 Написать программу для вычисления значения функции

$$y=\begin{cases}\sin x, & \text{при } x\le 0;\\ \operatorname{arctg} x, & \text{при } 0<x\le\dfrac{\pi}{4};\\ \log_2 x, & \text{при } \dfrac{\pi}{4}<x\le 32;\\ \dfrac{1}{x}, & \text{в остальных случаях;}\end{cases}$$

4 Написать программу для вычисления значения функции

$$y=\begin{cases}0, & \text{при } x\le 0;\\ \dfrac{1}{x}, & \text{при } 0<x\le 1;\\ x^2, & \text{при } 1<x\le 4;\\ 14+\log_2 x, & \text{в остальных случаях.}\end{cases}$$

5 Написать программу для вычисления значения функции.

$$z=\begin{cases}\operatorname{arctg}\dfrac{x}{y}, & \text{при } y\neq 0 \quad \text{и} \quad |x|>|y|;\\ \arcsin\dfrac{x}{y}, & \text{при } y\neq 0 \quad \text{и} \quad |x|\le|y|;\\ 0, & \text{в остальных случаях.}\end{cases}$$

6 Написать программу для вычисления значения функции.

$$y=\begin{cases}\dfrac{1}{x}+\dfrac{1}{y}, & \text{при } x<-10 \quad y<-5;\\ \dfrac{x-y}{x+y}, & \text{при } -10\le x<0 \text{ и } -5\le y<0;\\ \dfrac{\sin x}{\cos y}, & \text{при } 0\le x<2\pi \text{ и } 0\le y<\dfrac{\pi}{2};\\ \ln(x^2+y^2), & \text{в остальных случаях.}\end{cases}$$

7 Написать программу для вычисления значения функции

$$z=\begin{cases}x+y, & \text{при } x<0 \text{ и } y\le 0;\\ \operatorname{arctg}\dfrac{x}{y}, & \text{при } 0\le x<10 \text{ и } 0<y\le 8;\\ 0, & \text{в остальных случаях.}\end{cases}$$

8 Написать программу для вычисления значения функции.

$$y=\begin{cases}0, & \text{при } x<0;\\ \dfrac{1}{x^2+1}, & \text{при } 0\le x\le 1;\\ x^3, & \text{при } 1<x\le 4;\\ 62+\log_8 x, & \text{при } x>4.\end{cases}$$

9 Написать программу для вычисления значения функции

$$z=\begin{cases}\ln|x|, & \text{при } x<-\pi;\\ \sin x+\cos 2x, & \text{при } -\pi\le x<\pi;\\ x^3+1, & \text{при } \pi\le x<10;\\ \dfrac{x+1}{x^2+8}, & \text{при } 10\le x<100;\\ \ln x, & \text{в остальных случаях.}\end{cases}$$

10 Написать программу для вычисления значения функции

$$y=\begin{cases}\dfrac{2}{x}-\dfrac{4}{y}, & \text{при } x<-20 \quad y<-10;\\ \dfrac{x-y-2}{x+y}, & \text{при } -20\le x<0 \text{ и } -10\le x<0;\\ \dfrac{\sin x+\cos x}{\cos y}, & \text{при } 0\le x<2\pi \text{ и } 0\le x<\dfrac{\pi}{2};\\ \log_4(x^2+y^2), & \text{в остальных случаях.}\end{cases}$$

Задание 2

1 Даны три целых числа. Найти количество положительных чисел в исходном наборе.

2 Даны три целых числа. Найти количество положительных и количество отрицательных чисел в исходном наборе.

3 Даны две переменные вещественного типа: A, B. Перераспределить значения данных переменных так, чтобы в A оказалось меньшее из значений, а в B – большее. Вывести новые значения переменных A и B.

4 Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной сумму этих значений, а если равны, то присвоить переменным нулевые значения. Вывести новые значения переменных A и B.

5 Даны две целые переменные: A и B. Если их значения не равны, то присвоить каждой переменной большее из этих значений, а если равны, то присвоить переменным значение 0. Вывести новые значения переменных A и B.

6 Даны три числа. Найти наименьшее из них.

7 Даны три числа. Найти среднее из них (то есть число, расположенное между наименьшим и наибольшим).

8 Даны три числа. Вывести вначале меньшее, а затем большее из чисел.

9 Даны три числа. Найти сумму двух наибольших из них.

10 Даны три переменные вещественного типа: A, B, C. Если их значения упорядочены по возрастанию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных A, B, C.

11 Даны три переменные вещественного типа: A, B, C. Если их значения упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных A, B, C.

12 Даны три целых числа, одно из которых отлично от двух других, равных между собой. Определить порядковый номер числа, отличного от остальных.

4 Лабораторная работа № 4. Программирование с использованием оператора switch

Цель работы: получение практических навыков в работе с оператором switch.

Постановка задачи: написать программу для решения задач.

4.1 Основные сведения

Оператор выбора switch. Для выполнения многочисленных проверок в программах можно использовать конструкцию **if-else-if**, но она очень трудна для восприятия и в ней легко запутаться. С этой целью в языке С есть оператор принятия решений **switch**, выполняющий действия, основываясь на сравнении значения со списком констант символов или целых чисел. При обнаружении совпадения выполняется оператор или операторы, ассоциированные с данным значением. Оператор **switch** имеет следующий вид:

```
switch (выражение)  {
   case константа1 :
      последовательность операторов
  break;
   case константа2 :
      последовательность операторов
  break;
  . . .
   case константаN :
      последовательность операторов
  break;
   default :
      последовательность операторов
  break;
}
```

Оператор **default** выполняется, если не найдено соответствий. Оператор **default** не обязателен и в случае отсутствия совпадений ничего не происходит. Когда обнаруживается совпадение, операторы, ассоциированные с соответствующим **case**, выполняются до тех пор, пока не встретится оператор **break**. В случае **default** (или последнего **case**, если отсутствует **default**), оператор **switch** заканчивает работу при обнаружении конца.

Следует знать о трех важных моментах оператора switch.

1 Оператор **switch** отличается от оператора **if** тем, что он может выполнять только операции проверки строгого равенства, в то время как **if** может вычислять логические выражения и отношения.

2 Не может быть двух констант в одном операторе **switch**, имеющих одинаковое значение. Конечно, оператор **switch**, включающий в себя другой оператор **switch**, может содержать аналогичные константы.

3 Если в операторе **switch** используются символьные константы, они автоматически преобразуются к целочисленным значениям.

Пример

```
char sign;
int x, y, z;
switch (sign)  {
   case ‘–‘ :  z = x-y;
     break;
   case ‘+’ :  z = x+y;
     break;
   case ‘*’ :  z = x*y;
     break;
   case ‘/’ :  z = x/y;
   case ‘%’ : z = x%y;
     break;
   default : printf(“Неизвестная операция\n”);
}
```

С технической точки зрения операторы **break** являются необязательными в операторе **switch**. Они используются для окончания работы последовательности операторов, ассоциированных с данной константой. Если оператор **break**

отсутствует, продолжают выполняться операторы следующего раздела **case**, пока не будет достигнут оператор **break** или конец оператора **switch**. О константах выбора можно думать как о метках. Выполнение начинается с метки, соответствующей искомому значению, и будет продолжаться, пока не будет достигнут **break** или конец оператора **switch**. Можно использовать пустые условия.

Задание 1

1 Дан пол человека: м – мужчина, ж – женщина. Вывести на экран возможные мужские и женские имена в зависимости от введенного пола.

2 Дан признак транспортного средства: а – автомобиль, в – велосипед, м – мотоцикл, с – самолет, п – поезд. Вывести на экран максимальную скорость транспортного средства в зависимости от введенного признака.

3 Дан номер телевизионного канала. Вывести на экран наиболее популярные программы заданного канала.

4 Дан признак геометрической фигуры на плоскости: к – круг, п – прямоугольник, т – треугольник. Вывести на экран периметр и площадь заданной фигуры (данные, необходимые для расчетов, запросить у пользователя).

5 Дан номер масти m ($1 \le m \le 4$), определить название масти. Масти нумеруются: «пики» – 1, «трефы» – 2, «бубны» – 3, «червы» – 4.

6 Дан номер карты k ($6 \le k \le 14$), определить достоинство карты. Достоинства определяются по следующему правилу: «туз» – 14, «король» – 13, «дама» – 12, «валет» – 11, «десятка» – 10, …, «шестерка» – 6.

7 Написать алгоритм, позволяющий получить словесное наименование школьных оценок.

8 Написать алгоритм, который по номеру дня недели – целому числу от 1 до 7 выдавать в качестве результата количество пар в Вашей группе в соответствующий день.

9 Написать алгоритм нахождения числа дней в месяце, если даны: Номер месяца n – целое число a, равное 1 – для високосного года и равное 0 в противном случае.

10 В зависимости от того, введена ли открытая скобка или закрытая, напечатать "открытая круглая скобка" или "закрытая фигурная скобка" (учитывать круглые, квадратные, фигурные скобки).

11 В зависимости от введённого символа L, S, V программа должна вычислять длину окружности; площадь круга; объём цилиндра.

12 Написать программу, которая по введенному числу из промежутка 1…12 определяет пору года.

Задание 2

В старояпонском календаре принят 60-летний цикл, состоящий из пяти 12-летних подциклов. Подциклы обозначаются названиями цвета: зеленый, красный, желтый, белый и черный. Внутри каждого подцикла годы носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. (1924 год – *год зеленой крысы* – был нача-

лом очередного цикла). Написать программу, которая вводит номер некоторого года и печатает его название по старояпонскому календарю.

5 Лабораторная работа № 5. Оператор цикла for. Логические и поразрядные операции

Цель работы:
– изучение циклических операторов языка С++;
– получение навыков в программирование цикла с параметром for.

Постановка задачи: разработать программы циклических процессов:
- вывести таблицу значений функции на заданном отрезке;
- вычислить значение конечной суммы или произведения.

5.1 Общие сведения

Операторы цикла используются для организации многократно повторяющихся вычислений. Любой цикл состоит из тела цикла, то есть тех операторов, которые выполняются несколько раз, начальных установок, модификации *параметра* цикла и проверки условия продолжения выполнения цикла. Один проход цикла называется итерацией. Все циклы в С++ выполняют тело цикла, пока условие истинно.

Стандартный вид цикла **for** следующий:

for (*инициализация*; *условие*; *модификация*)
***тело_цикла*;**

Оператор **for** имеет три главные части.

1 Инициализация – оператор присваивания, используемый для установки начального значения переменной цикла.

2 Условие – выражение, определяющее условие работы цикла.

3 Модификация – выражение, определяющее характер изменения переменной цикла на каждой итерации.

Эти три важные части должны разделяться точкой с запятой. Цикл **for** работает до тех пор, пока условие истинно. Когда условие становится ложным, выполнение программы продолжается с оператора, следующего за циклом **for**.

Порядок выполнения: переменной *счётчика цикла* присваивается начальное значение (инициализация) и проверяется условие; если условие неверно, то *тело цикла* не выполняется и управление передается на оператор, следующий за конструкцией **for**. Если же условие выполняется, то выполняется *тело цикла*, затем изменяется значение *счётчика цикла* и снова проверяется условие. Данный процесс будет выполняться, пока условие не станет ложным**.**

Рассмотрим принцип работы оператора на примере, где осуществляется вывод чисел от 1 до 4 включительно (рисунок 5.1).

В данной программе переменная **x** изначально установлена в 1. Поскольку **x** меньше 4, выводится с помощью cout значение 1, после чего **x** увеличива-

ется на 1 и проверяется условие: по-прежнему ли **x** меньше либо равно 4. Данный процесс продолжается до тех пор, пока **x** не станет больше 4, и в этот момент цикл прервется. В данном примере **x** является *переменной цикла*, которая изменяется и проверяется на каждой итерации цикла.

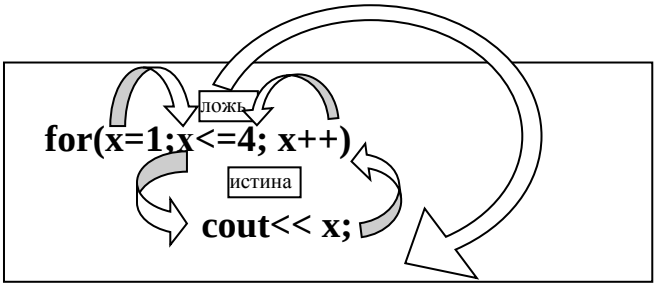


Рисунок 5.1 – Принцип работы оператора на примере

Изображение в блок-схемах (рисунок 5.2).

Рисунок 5.2 – Принцип работы оператора на примере

При написании программы с использованием оператора цикла **for** обратите внимание на следующие моменты:

– если тело цикла состоит более чем из одного оператора, то они заключаются в операторные скобки;

– после условий цикла точка с запятой не ставится (если тело цикла не пустой оператор);

– одна или все из частей оператора for могут отсутствовать, но точки с запятой надо оставить на своих местах.

Пример – Построить таблицу значений для функции

$$f(x)=\begin{cases}\sqrt{|ax|}, & \text{если } x\le 0 \text{ и } a=b;\\ a\sin bx, & \text{если } 0<x\le b \text{ или } b>3;\\ 0, & \text{в остальных случаях.}\end{cases}$$

Функция f(x) должна принимать действительное значение, если выражение Ац И (НЕ Вц ИЛИ Ац) не равно нулю, и целое значение – в противном случае. Через Ац и Вц обозначены целые части значений a, b, операции И, НЕ и ИЛИ – битовые. Значения a, b, $X_{нач}$, $X_{кон}$, dX ввести с клавиатуры.

Исходными данными являются начальное значение аргумента Xn, конечное значение аргумента Xk, шаг изменения аргумента dX и параметры a, b, c. Все величины – вещественные. Программа должна выводить таблицу, состоящую из двух

столбцов – значений аргумента x и соответствующих им значений функции f.

В программе будут использованы, кроме логических операций (И – &&, ИЛИ – ||), битовые (И – &, ИЛИ – |, НЕ – ~), которые работают с битами двоичного представления целых чисел. Для выделения целых частей числа используется операция преобразования типов int().

Текст программы:

```
        //директивы препроцессора
#include <iostream.h>    //для использования cin, cout
#include <math.h>        // для использования sqrt(), sin(), fabs()
#include <windows.h>   //для использования CharToOem()
char  buf[256];
char* RUS(const char* text) //функция для вывода кириллицы в консоли
{
  CharToOem(text, buf);
  return buf;}
int main()
{
  float a,b,Xn,Xk,dX,x,f;   //объявление переменных вещественного типа
  //вывод приглашения к вводу
cout<<RUS("Введите коэффициенты a и b:")<<endl;
  cin>>a>>b;  //ввод исходных данных
  //вывод приглашения к вводу
cout<<RUS("Введите Хнач, Хкон и шаг dX:")<<endl;
  cin>>Xn>>Xk>>dX;  //ввод исходных данных
  for(x=Xn;x<=Xk;x+=dX)  //условия цикла
  {
        if(x<=0 && a==b)
              f=sqrt(fabs(a*x));
        else if((x>0 && x<=b) || b>3)
              f=a*sin(b*x);
        else
              f=0;
        //вывод в зависимости от условия
        if(int(a)&(~int(b)|int(a))!=0)
              cout<<"x= "<<x<<", f(x)= "<<f<<endl;
        else
              cout<<"x= "<<x<<", f(x)= "<<int(f)<<endl;
  }
  return 0;}
```

Задание 1

Циклический вычислительный процесс табулирование функции.

1 Построить таблицу значений для функции $f(x) = x - \sin(x)$ на отрезке $[0; \pi/2]$ с числом разбиений отрезка m = 10.

2 Построить таблицу значений для функции $f(x) = \sin(x)$ на отрезке $[\pi/4; \pi/2]$ с числом разбиений отрезка m = 15.

3 Построить таблицу значений для функции $f(x) = \cos(x)$ на отрезке $[\pi/3; 2\pi/3]$ с числом разбиений отрезка m = 20.

4 Построить таблицу значений для функции $f(x) = tg(x)$ на отрезке $[0; \pi/4]$ с числом разбиений отрезка m = 10.

5 Построить таблицу значений для функции $f(x) = ctg(x)$ на отрезке $[\pi/4; \pi/2]$ с числом разбиений отрезка m = 15.

6 Построить таблицу значений для функции $f(x) = \arcsin(x)$ на отрезке [0; 1] с числом разбиений отрезка m = 20.

7 Построить таблицу значений для функции $f(x) = \arccos(x)$ на отрезке [0.5; 1] с числом разбиений отрезка m = 10.

8 Построить таблицу значений для функции $f(x) = arctg(x)$ на отрезке [2; 7] с числом разбиений отрезка m = 15.

9 Построить таблицу значений для функции $f(x) = \sin(x) - \cos(x)$ на отрезке $[0; \pi/2]$ с числом разбиений отрезка m = 20.

10 Построить таблицу значений для функции $f(x) = x\sin(x)$ на отрезке $[0; 3\pi]$ с числом разбиений отрезка m = 10.

11 Построить таблицу значений для функции $f(x) = \sin\left(\frac{1}{x}\right)$ на отрезке $[\pi/8; 2/\pi]$ с числом разбиений отрезка m = 10.

Задание 2

Циклический вычислительный процесс: конечные суммы и произведения.

1 Вычислить значение конечной суммы: $\frac{\sin x}{1} + \frac{\sin 2x}{2} + \cdots + \frac{\sin nx}{n}$.

2 Вычислить значение конечной суммы: $\frac{1}{(1+1)^2} + \frac{1}{4(2+1)^2} + \cdots + \frac{1}{n^2(n+1)^2}$.

3 Вычислить значение конечной суммы: $\frac{1}{1!} + \frac{4}{2!} + \cdots + \frac{n^2}{n!}$.

4 Вычислить значение конечной суммы: $\frac{1}{(2+1)} + \frac{1}{2(4+1)} + \cdots + \frac{1}{n(2n+1)}$.

5 Вычислить значение конечной суммы: $\frac{1}{(1+1)^2} + \frac{3}{4(2+1)^2} + \cdots + \frac{2n-1}{n^2(n+1)^2}$.

6 Вычислить значение конечной суммы: $\frac{1}{3} + \frac{1}{8} + \cdots + \frac{1}{n^2-1}$.

7 Вычислить значение конечного произведения: $\left(1+\frac{1}{1(1+2)}\right)\cdot\left(1+\frac{1}{2(2+2)}\right)\cdots\left(1+\frac{1}{n(n+2)}\right)$.

8 Вычислить значение конечного произведения: $\frac{5}{8}\cdot\frac{12}{15}\cdot\ldots\cdot\frac{n^2-4}{n^2-1}$.

9 Вычислить значение конечного произведения: $\cos\frac{x}{2}\cdot\cos\frac{x}{2^2}\cdot\ldots\cdot\cos\frac{x}{2^n}$.

10 Вычислить значение конечного произведения: $\cos\frac{\pi}{2^2}\cdot\cos\frac{\pi}{2^3}\cdot\ldots\cdot\cos\frac{\pi}{2^{n+1}}$.

11 Вычислить значение конечного произведения: $(1+1)\cdot\left(1+\left(\frac{1}{2}\right)^2\right)\cdot\ldots\cdot\left(1+\left(\frac{1}{2}\right)^{2n}\right)$.

12 Вычислить значение конечного произведения: $(1+1)(1+x^2)\cdot\ldots\cdot(1+x^{2n})$.

6 Лабораторная работа № 6. Операторы цикла while и do while. Нахождение бесконечных сумм и произведений

Цель работы:

– изучение циклических операторов языка С++;

– получение навыков в программирование циклических алгоритмов while, do ... while;

– изучение операторов перехода.

Постановка задачи: разработать программы циклических процессов для вычисления значений бесконечной суммы или произведения.

6.1 Общие сведения

Цикл с предусловием имеет вид:

while (условие) тело_цикла;

Условие определяет условие повторения тела цикла, представленного простым или составным оператором. Выполнение оператора начинается с вычисления условия. Если оно истинно (не равно false), выполняется тело цикла. Если при первой проверке выражение равно false, цикл не выполнится ни разу. Тип условия должен быть арифметическим или приводимым к нему. Условие вычисляется перед каждой итерацией цикла.

Пример – Найти сумму цифр введенного целого числа.

Исходными данными является целое число n, выходными данными будет сумма цифр – целое число sum.

Для выделения цифр в числе воспользуемся следующим алгоритмом (рисунок 6.1):

1) остаток от деления на 10 дает последнюю цифру числа;

2) при делении на 10 последняя цифра числа отбрасывается;

3) будем выполнять действия 1–2 и находить сумму выделяемых цифр, пока в числе не будут отброшены все цифры, т. е. пока число будет больше нуля.

В противоположность циклам for и while, сначала проверяющим условие, цикл do…while проверяет условие в конце. То есть цикл do while всегда выполняется, по крайней мере, один раз. Стандартный вид цикла do while следующий:

```
do
{
  тело_цикла;
}
while (условие);
```

Сначала выполняется простой или составной оператор, составляющий тело цикла, а затем вычисляется выражение. Если оно истинно (не равно false), тело цикла выполняется еще раз. Цикл завершается, когда выражение станет равным false или в теле цикла будет выполнен какой-либо оператор перехода, пример схемы алгортма цикла представлен на рисунке 6.1.

Текст программы:

```
#include <iostream.h>
#include <windows.h>
char  buf[256];
char* RUS(const char* text)
{
  CharToOem(text, buf);
  return buf;}
int main()
{
        int n,sum;
        cout<<RUS("Введите число:")<<endl;
        cin>>n;
        sum=0;
        while (n>0) //условие цикла
        {
                //увеличиваем sum на последнюю
                //цифру
                sum+=n%10;
                //отбрасываем последнюю цифру
                n/=10;
        }
        cout<<RUS("Сумма цифр: ")<<sum<<endl;
        return 0; }
```

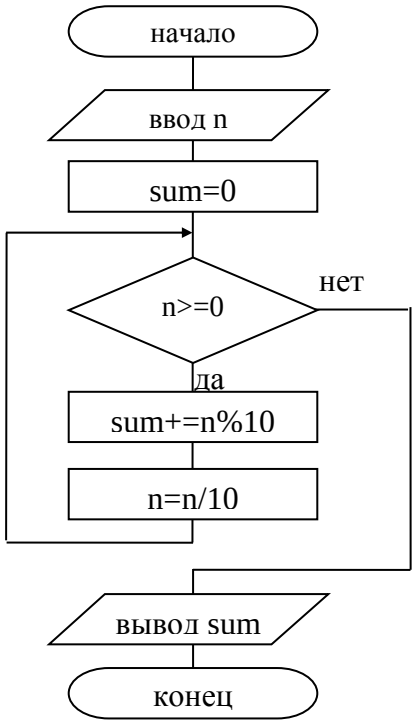


Рисунок 6.1 – Схема алгоритма программы

Изображение **do…while** в блок-схемах представлено на рисунке 6.2.

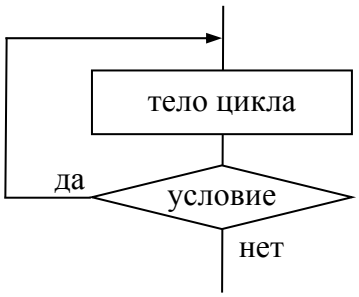


Рисунок 6.2 – Изображение do…while в блок-схемах

Пример 2 **–** Написать программу вычисления значения функции cosx с помощью бесконечного ряда Тейлора с точностью ε по формуле

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots + (-1)^n \frac{x^{2n}}{2n!} + \cdots .$$

Для достижения заданной точности необходимо суммировать члены ряда до тех пор, пока очередной член по модулю не будет меньшим ε.

Воспользуемся рекуррентной формулой для получения последующего члена ряда через предыдущий:

$$C_{n+1} = T \cdot C_n,$$

где T – некоторый множитель, который находится следующим образом:

$$C_n = \frac{(-1)^n x^{2n}}{(2n)!}; \qquad C_{n+1} = \frac{(-1)^{n+1} x^{2(n+1)}}{(2(n+1))!}; \; T = \frac{C_{n+1}}{C_n} = \frac{(-1)^{n+1} x^{2(n+1)} (2n)!}{(-1)^n x^{2n} (2(n+1))!} = -\frac{x^2}{(2n+1)(2n+2)}.$$

Для решения воспользуемся следующим алгоритмом (рисунок 6.3).

Текст программы.

```
#include <iostream.h>
#include <windows.h>
#include<math.h>
char  buf[256];
char* RUS(const char* text)
{   CharToOem(text, buf);    return buf;}
int main(void)
{
  const int MaxIter=100;
  double x,eps;
  double Cn,y;// член ряда и  сумма.
  int n; // количество итераций.
  cout<<RUS("Введите x и точность:");
  cin>>x>>eps;
  y=Cn=1;
  n=0;
  do
   {
    //очередной член ряда
    Cn*=-x*x/((2*n+1)*(2*n+2));
    y+=Cn;//вычисление суммы
    n++;
    if(n>MaxIter)
     { cout<<RUS("Ряд расходится.");
       break;//оператор выхода из цикла
     }
   }
  while(fabs(Cn)>eps);
  if(fabs(Cn)<=eps)
  {
    cout<<RUS("cos x=: ") <<y<<endl;
    cout<<RUS("Для проверки:");
    cout<<cos(x)<<endl;   }
  return 0;}
```

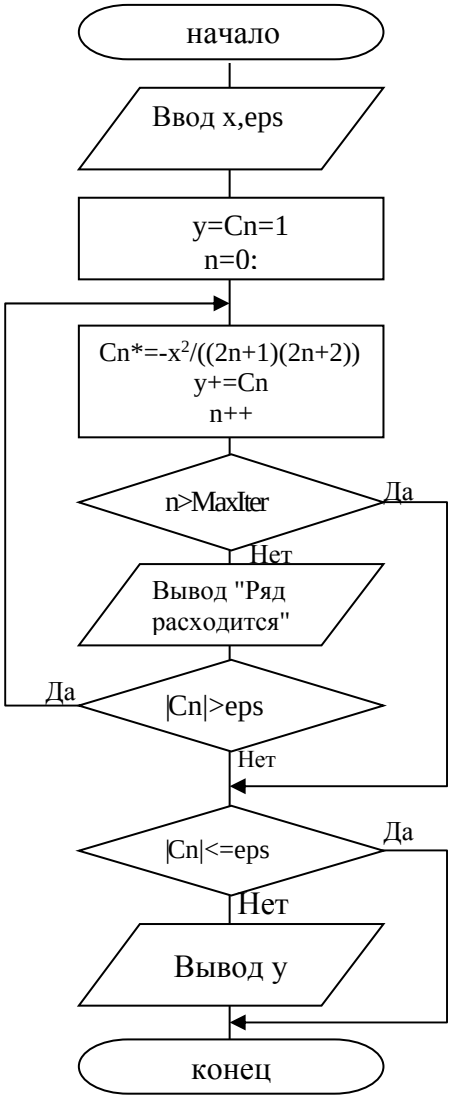


Рисунок 6.3 – Схема алгоритма программы

Задание 1

1 Вводить последовательность чисел до тех пор, пока их сумма не достигнет M (M вводится и больше 0). Ввести, какое количество чисел составили искомую сумму (саму сумму тоже).

2 Вводить последовательность до тех пор, пока не встретятся три подряд

идущих положительных числа. Тогда прервать ввод и сообщить, сколько во введенной последовательности было:

а) всего чисел;

б) положительных чисел;

в) отрицательных чисел.

3 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,0001. $1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^n}+\cdots$.

4 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,05. $1-\frac{1}{2}+\frac{1}{4}-\frac{1}{8}+\ldots\pm\frac{1}{2^n}\mp\ldots$.

5 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,00005. $\frac{1}{1\cdot3}+\frac{1}{3\cdot5}+\cdots+\frac{1}{(2n-1)(2n+1)}+\cdots$.

6 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,0001. $\frac{1}{3\cdot5}+\frac{1}{7\cdot9}+\cdots+\frac{1}{(4n-1)(4n+1)}+\cdots$.

7 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,001. $\frac{1}{1^2}+\frac{1}{3^2}+\cdots+\frac{1}{(2n+1)^2}+\cdots$.

8 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,005. $1+\frac{1}{2^4}+\frac{1}{3^4}+\cdots+\frac{1}{n^4}+\cdots$.

9 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,0005. $1-\frac{1}{2^4}+\frac{1}{3^4}-\ldots\pm\frac{1}{n^4}\mp\ldots$.

10 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,01. $\frac{1}{1^4}+\frac{1}{3^4}+\cdots+\frac{1}{(2n+1)^4}+\cdots$.

11 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,05. $\frac{1}{1\cdot4}+\frac{1}{4\cdot7}+\cdots+\frac{1}{(3n-2)(3n+1)}+\cdots$.

Задание 2

Вычислить и вывести на экран в виде таблицы значение функции, заданной с помощью ряда Тейлора, на интервале от $X_{нач}$ до $X_{кон}$ с шагом dX с точностью ε. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда. Для вычисления последующего члена ряда использовать рекуррентную формулу:

1) $e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}+\ldots,\quad |x|<\infty$;

2) $e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \ldots + (-1)^n \frac{x^n}{n!} + \ldots, \quad |x| < \infty$;

3) $\ln \frac{x+1}{x-1} = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = 2(1 + \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots \quad |x| > 1$;

4) $\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots, \quad |x| < \infty$;

5) $\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} - \cdots, \quad |x| < \infty$;

6) $\ln(x+1) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{n+1}}{n+1} = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots \quad -1 < x \le 1$;

7) $\ln \frac{1+x}{1-x} = 2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = 2(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots, \quad |x| < 1$;

8) $\operatorname{arcctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} (-1)^{n+1} \frac{x^{2n+1}}{2n+1} = \frac{\pi}{2} - \frac{x}{1} + \frac{x^3}{3} - \frac{x^5}{5} + \cdots, \quad |x| \le 1$;

9) $\operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \cdots, \quad |x| \le 1$;

10) $\ln(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n} = -\left(\frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \cdots \right), \quad -1 \le x < 1$;

11) $\operatorname{arctg} x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1} = \frac{x}{1} - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots, \quad |x| \le 1$;

12) $\operatorname{arctg} x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \cdots, \quad x < -1$.

7 Лабораторная работа № 7. Обработка одномерных массивов

Цель работы:
- получение практических навыков в работе с одномерными массивами;
- ознакомление с алгоритмами упорядочения.

Постановка задачи: для конкретного варианта ввести массив исходных данных и выполнить над ним указанные действия. Изучив алгоритмы упорядочения, выбрать один из них. Написать программу, которая работает с любым набором данных.

7.1 Общие сведения

Массив – это совокупность переменных одного типа, к которым обращаются с помощью общего имени. Доступ к отдельному элементу массива может осуществляться с помощью индекса. В языке C++ все массивы состоят из со-

прикасающихся участков памяти. Наименьший адрес соответствует первому элементу, наибольший адрес соответствует последнему элементу. Массивы могут иметь одну или несколько размерностей.

Одномерный массив.

Стандартный тип объявления одномерного массива следующий:

тип имя_массива[*размер*];

В C++ массивы должны определяться однозначно, чтобы компилятор мог выделить для них место в памяти. Здесь *тип* объявляет базовый тип массива и является типом каждого элемента массива. Параметр *размер* определяет, сколько элементов содержит массив, и может быть целым числом или целочисленной именованной константой, но не переменной.

У всех массивов первый элемент имеет индекс 0. Поэтому если написать *int p*[*10*];, то будет объявлен массив целых чисел из 10 элементов, причем эти элементы адресуются индексом от 0 до 9.

Для доступа к элементу массива используется следующий синтаксис:

имя_массива[*индекс*]

Например, последнему элементу массива присвоим значение первого элемента: *p*[*9*]=*p*[*0*];

Для работы с массивами используются циклы. Следующий фрагмент программы вводит целочисленный массив с клавиатуры и выводит его на дисплей:

```
const int n=10;
int x[n]; /* резервирует место для 10 целочисленных элементов */
int i;
//ввод элементов массива
for (i=0; i<n; i++)
{
  cout<<RUS(“Введите “)<< i+1; cout<<RUS(“-й элемент массива: “);
  cin>>x[i];
}
//вывод массива на экран
for (i=0; i<n; i++)
  cout<< x[i]<<”\t”;
```

В языке С отсутствует проверка границ массивов. Можно выйти за один конец массива и записать значение в какую-либо переменную, не относящуюся к массиву, или даже в код программы. Работа по проверке границ массива возлагается на программиста.

Пример – В последовательности действительных чисел найти количество положительных элементов и обменять минимальный элемент с первым.

Для поиска количества положительных элементов используется следующий алгоритм: просматриваем поочередно все элементы и если элемент массива больше нуля, увеличить счетчик элементов на единицу.

Чтобы поменять местами минимальный и первый элемент массива, необходимо найти местоположения минимального элемента, то есть его индекс. Для

этого нужно провести следующие операции.

1 Задать начальные значения для индекса минимального элемента (например, 0 или другие значения индекса, не выходящие за границу массива).

2 Просмотреть массив, поочередно сравнивая каждый его элемент с ранее найденным минимумом. Если очередной элемент меньше ранее найденного минимума, принять этот элемент за новый минимум (запомнить его индекс).

Для обмена необходимо использовать дополнительную переменную. Процесс обмена проиллюстрируем на рисунке 7.1, а алгоритм – на рисунке 7.2.

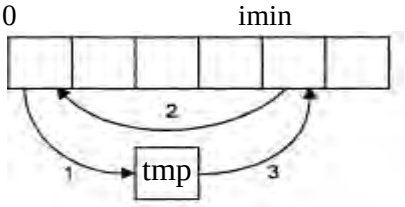


Рисунок 7.1 – Процесс обмена элементов массива

Текст программы:

```
#include <iostream.h>
#include <windows.h>
char  buf[256];
char* RUS(const char* text)
{   CharToOem(text, buf);
  return buf; }
int main(void)
{  const int n=7;   //размерность массива
float b[n];          // описание массива
  float tmp;
  int i,imin;
  int pol;
  // ввод массива
  cout<<RUS("Введите элементы массива:");
  for (i = 0; i<n; i++)
 cin >> b[i];
//подсчет количества положительных
pol=0;
for (i = 0; i<n; i++)
  if (b[i]>0)
    pol++;
// принимаем за наименьший первый из элементов
  imin=0;
  for (i = 0; i<n; i++)
// если нашли меньший элемент
    if(b[i]<b[imin])
    imin=i; запоминаем его номер
 // обмен элементов b[0] и b[imin]:
  tmp=b[0];                //1
  b[0]=b[imin]; //2
  b[imin]=tmp; //3
  //вывод полученного массива
  for (i = 0; i<n; i++)
```

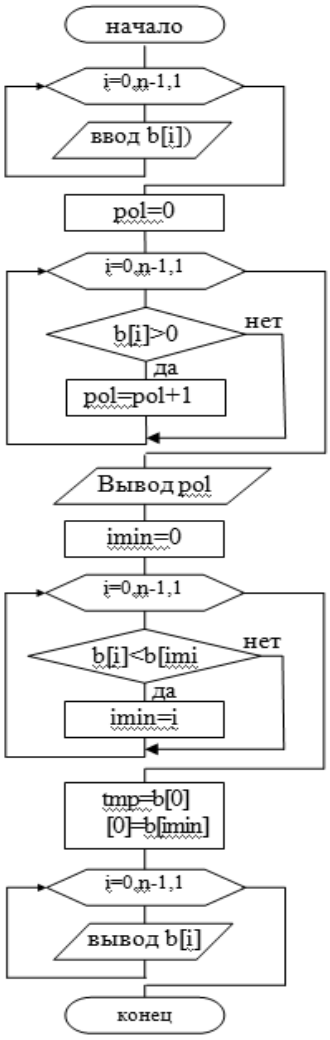


Рисунок 7.2 – Схема алгоритма программы

```
  cout<<b[i]<<"\t";
cout<<endl;
return 0;}
```

Сортировка.

Рассмотрим массив целых или вещественных чисел a_1, a_2, …, a_n. Пусть требуется переставить элементы этого массива так, чтобы после перестановки они были упорядочены по неубыванию $a_1 \le a_2 \le \ldots \le a_n$ или по невозрастанию $a_1 \ge a_2 \ge \ldots \ge a_n$. Эта задача называется задачей сортировки или упорядочения массива. Существуют различные алгоритмы:

1) найти элемент массива, имеющий наименьшее (наибольшее) значение, переставить его с первым элементом. Затем проделать то же самое, начав со второго элемента, и так далее (сортировка выбором);

2) последовательным просмотром чисел a_1, a_2, …, a_n найти наименьшее i такое, что $a_i > a_{i+1}$ или $a_i < a_{i+1}$. Поменять a_i и a_{i+1} местами, возобновить просмотр с элемента a_{i+1} и так далее. Тем самым самое наибольшее или наименьшее число передвинется на последнее место. Следующие просмотры следует начинать опять сначала, уменьшая на единицу количество просматриваемых элементов. Массив будет упорядочен после просмотра, в котором участвовали только первый и второй элементы (сортировка обменами);

3) просматривать последовательно a_2, …, a_n и каждый новый элемент вставлять на подходящее место в уже упорядоченную последовательность a_1, …, a_{i-1}. Это место определяется последовательным сравнением a_i с упорядоченными элементами a_1, …, a_{i-1} (сортировка простыми вставками);

4) сравнить элементы a_1 и a_2 и, если $a_1 > a_2$ (или $a_1 < a_2$), то эти элементы переставить. Затем сравнить элементы a_2 и a_3 и, если $a_2 > a_3$ (или $a_2 < a_3$), то их переставить. После сравнить элементы a_3 и a_4 и так далее до элементов a_{n-1} и a_n включительно. Затем эти действия повторить, начиная опять с первого элемента. Последним является контрольный проход, при котором не будет перестановок элементов (сортировка по методу пузырька).

Задание

1 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) сумму отрицательных элементов массива;

б) произведение элементов массива, расположенных между максимальным и минимальным элементами. Упорядочить элементы массива по возрастанию.

2 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) сумму положительных элементов массива;

б) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами. Упорядочить элементы массива по убыванию.

3 В одномерном массиве, состоящем из n целых элементов, вычислить:

а) произведение элементов массива с четными номерами;

б) сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все

положительные элементы, а потом — все отрицательные (элементы, равные 0, считать положительными).

4 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) сумму элементов массива с нечетными номерами;

б) сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

5 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) максимальный элемент массива;

б) сумму элементов массива, расположенных до последнего положительного элемента.

Сжать массив, удалив из него все элементы, модуль которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

6 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) минимальный элемент массива;

б) сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом — все остальные.

7 В одномерном массиве, состоящем из n целых элементов, вычислить:

а) номер максимального элемента массива;

б) произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях.

8 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) номер минимального элемента массива;

б) сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом — все остальные.

9 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) максимальный по модулю элемент массива;

б) сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

10 В одномерном массиве, состоящем из n целых элементов, вычислить:

а) минимальный по модулю элемент массива;

б) сумму модулей элементов массива, расположенных после первого элемента, равного нулю. Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

11 В одномерном массиве, состоящем из n вещественных элемен-

тов, вычислить:

а) номер минимального по модулю элемента массива;

б) сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

12 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) номер максимального по модулю элемента массива;

б) сумму элементов массива, расположенных после первого положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [a, b], а потом – все остальные.

13 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) количество элементов массива, лежащих в диапазоне от A до B;

б) сумму элементов массива, расположенных после максимального элемента.

Упорядочить элементы массива по убыванию модулей элементов.

14 В одномерном массиве, состоящем из n вещественных элементов, вычислить:

а) количество элементов массива, равных 0;

б) сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей элементов.

8 Лабораторная работа № 8. Обработка двумерных массивов

Цель работы:

- изучение двумерных массивов;
- закрепление навыков структурного программирования.

Постановка задачи: для конкретного варианта ввести двумерный массив и выполнить над ним указанные действия.

8.1 Основные сведения

Двумерные массивы.

Язык C++ позволяет создавать многомерные массивы. Простейшим видом многомерного массива является двумерный массив. Двумерный массив – это массив одномерных массивов. Двумерный массив объявляется следующим образом:

тип имя_массива[*размер1*][*размер2*];

Следовательно, для объявления двумерного массива целых чисел с размером 5 на 8 следует написать:

```
int  d[5][8];
```

В противоположность другим языкам программирования, где размерности массива отделяются запятой, язык C++ помещает каждую размерность в отдельные скобки.

Для доступа к элементу многомерного массива указываются все его индексы, например, для доступа к элементу в третьей строке пятого столбца массива d следует использовать d[2][4] (не забываем, что в массивах индексация начинается с нуля).

В следующем примере вводится по строкам двумерный массив и затем выводится построчно на экран:

```
int main()
{
  int  i, j;
  int num[3][4];
  for (i=0; i<3; i++)
  {
     cout<<RUS("Введите элементы")<<i+1;
          cout<<RUS("строки: \n");
     for (j=0; j<4; j++)
        cin>>num[i][j];
   }
  for (i=0; i<3; i++)
  {
    for (j=0; j<4; j++)
       cout<<num[i][j]<<"\t";
   cout<<endl;
  }
  return 0; }
```

Двумерные массивы можно представить в виде матрицы, где первый индекс отвечает за строку, а второй – за столбец. Это означает, что правый индекс изменяется быстрее левого, если двигаться по массиву в порядке расположения элементов в памяти.

Пример **–** Написать программу, которая определяет в целочисленной матрице номер строки, содержащей наибольшее количество элементов, равных нулю.

Исходными данными будет являться матрица b (точнее ее элементы).

Результат программы – номер строки с наибольшим количеством нулей istr.

Для поиска строки с наибольшим количеством нулей будем использовать следующий алгоритм: для каждой строки будем находить количество нулей Kol и сравнивать с максимальным количеством нулей MaxKol. Если значение Kol больше MaxKol, то в переменной MaxKol запоминаем количество нулей в данной строке и запоминаем номер строки.

Текст программы:

```
// размерности массива
const int nstr = 4, nstb = 5;
// описание двумерного массива
int b[nstr][nstb];
int  i, j;
int istr = -1, MaxKol = 0, Kol;
//ввод  массива
```

```
for (i=0; i<nstr; i++)
{
  cout<<RUS("Введите элементы ")<<i+1;
  cout<<RUS(" строки: \n");
  for (j=0; j<nstb; j++)
    cin>>b[i][j];
}
istr = -1; MaxKol =0;
for (i = 0;i<nstr; i++)
{ // просмотр массива по строкам
  Kol = 0;
  for (j = 0;j<nstb;j++)
        if (b[i][j] == 0)
          Kol++;
  if (Kol > MaxKol)
  {     istr = i;
        MaxKol = Kol;
  }
}
cout<<RUS("Исходная матрица:")<<endl;
 for (i=0; i<nstr; i++)
{
   for (j=0; j<nstb; j++)
      cout<<b[i][j]<<"\t";
    cout<<endl;
}
if(istr==-1)
cout<<RUS("Нулевых элементов нет.")<<endl;
else
  cout<<RUS("Номер строки с наибольшим количеством нулей: ")<<istr+1<<endl;
```

Задание 1

Исходные данные должны включать и положительные числа, и отрицательные числа, и нули. Массив заполнить случайными числами.

1 Дана целочисленная прямоугольная матрица. Определить:

а) количество строк, не содержащих ни одного нулевого элемента;

б) максимальное из чисел, встречающихся в заданной матрице более одного раза.

2 Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих ни одного нулевого элемента. Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

3 Дана целочисленная прямоугольная матрица. Определить:

а) количество столбцов, содержащих хотя бы один нулевой элемент;

б) номер строки, в которой находится самая длинная серия одинаковых элементов.

4 Дана целочисленная квадратная матрица. Определить:

а) произведение элементов в тех строках, которые не содержат отрицательных элементов;

б) максимум среди сумм элементов диагоналей, параллельных глав-

ной диагонали матрицы.

5 Дана целочисленная квадратная матрица. Определить:

а) сумму элементов в тех столбцах, которые не содержат отрицательных элементов;

б) минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

6 Дана целочисленная прямоугольная матрица. Определить:

а) сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент;

б) номера строк и столбцов всех седловых точек матрицы.

Примечание – Матрица A имеет седловую точку A_{ij}, если A_{ij} является минимальным элементом в i-й строке и максимальным в j-м столбце.

7 Для заданной матрицы размером 8 × 8 найти такие k, что k-я строка матрицы совпадает с k-м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

8 Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик. Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

9 Соседями элемента A_{ij} в матрице назовем элементы A_{kl} с $i - 1 \le k \le i + 1$, $j - 1 \le l \le j + 1$, $(k, l) \ne (i, j)$. Операция сглаживания матрицы дает новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы размером 10 × 10. В сглаженной матрице найти сумму модулей элементов, расположенных ниже главной диагонали.

10 Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Подсчитать количество локальных минимумов заданной матрицы размером 10 × 10. Найти сумму модулей элементов, расположенных выше главной диагонали.

11 Коэффициенты системы линейных уравнений заданы в виде прямоугольной матрицы. С помощью допустимых преобразований привести систему к треугольному виду. Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.

12 Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.

13 Осуществить циклический сдвиг элементов прямоугольной матрицы на n элементов вправо или вниз (в зависимости от введенного режима). n может быть больше количества элементов в строке или в столбце.

14 Осуществить циклический сдвиг элементов прямоугольной матрицы размерности M × N вправо на k элементов следующим образом: элементы 1-й строки сдвигаются в последний столбец сверху вниз, из него – в последнюю строку справа налево, из нее – в первый столбец снизу вверх, из него – в первую строку; для остальных элементов – аналогично.

9 Лабораторная работа № 9. Обработка строк

Цель работы:
- изучение строковых типов;
- использование строковых типов.

Постановка задачи: для конкретного варианта реализовать задачи с использованием типов char, string.

9.1 Основные сведения

Строка представляет собой массив символов, заканчивающийся нуль-символом. Нуль-символ — это символ с кодом, равным 0, что записывается в виде управляющей последовательности '\0'. По положению нуль-символа определяется фактическая длина строки. Строку можно инициализировать строковым литералом :

```
char str[10] = "Vasia"; // выделено 10 элементов с номерами от 0 до 9
                        // первые элементы - 'V .  'a',  's',  ' i',  'a',  '\0'
```

При вводе строк функция *cin* считывает последовательность введенных символов до первого пробела. Поэтому для ввода строк состоящих из нескольких слов, используется метод *getline* объекта *cin*, который считывает n-1 или менее символов и записывает их в строковую переменную.

Пример использования:

```
const int n=80;
char str[n];  //объявление  строки
cin.getline(str,n);  //ввод строки
cout<<s<<endl;  //вывод строки
```

Функции работы со строками представлены в таблице 9.1. Они находятся в библиотеке string.h.

Таблица 9.1 – Функции работы со строками

Действие	Вид функции	Пример использования
1	2	3
1 Присваивание строк	**char *strcpy (char *str1, const char *str2)** используется для копирования содержимого *str2* в *str1*	Следующий фрагмент кода копирует “hello” в строку *str*: char str[80]; strcpy(str,”hello”);
2 Длина строки	**size_t strlen (const char *str)** возвращает длину строки. При определении длины строки нулевой символ не учитывается	Данный фрагмент кода выводит на экран число 5. strcpy(s,”hello”); int n; n=strlen(s); cout<<n;

Окончание таблицы 9.1

1	2	3

3 Функция соединения строк	**char *strcat (char *str1, const char *str2)** конкатенирует (соединяет в цепочку) строку *str1* и копию строки *str2*. Строка *str2* остается в первоначальном виде	#include <iostream.h>
		#include <string.h>
		int main(void)
		{ char s1[80],s2[80];
		cin.getline(s1,80);
		cin.getline(s2,80);
		strcat(s2,s1);
		cout<<s2<<endl;
		return 0;}

4 Сравнение строк	**char *strcmp (char *str1, char *str2)** выполняет алфавитное сравнение двух строк и возвращает целое число со следующим значением:
	Число **Значение**
	Меньше 0 *str1* меньше, чем *str2*
	Равно 0 *str1* равна *str2*
	Больше 0 *str1* больше, чем *str2*
		Следующая функция используется для проверки пароля
		char s[80];
		cout<<“Введи пароль: “;
		cin.getline(s,80);
		if(strcmp(s, “pass”))
		cout<<“Неправильный пароль.”;

5 Поиск подстроки в строке	**char *strstr (char *str1, char *str2)** возвращает указатель на первое вхождение в строку, на которую указывает *str1*, строки, указанной *str2* . Если совпадений не обнаружено, возвращается нулевой указатель **NULL**	char *p;
		char s1[80],s2[80];
		cin.getline(s1,80);
		cin.getline(s2,80);
		p = strstr(s1,s2)
		if(p) cout<<”Входит s2 в s1”;
		else cout<<”Не входит s2 в s1”;

6 Поиск символа в строке	**char *strchr (const char *str, int ch)** возвращает указатель на первое вхождение символа *ch* в строку, на которую указывает *str*. Если символ *ch* не найден, возвращается нулевой указатель **NULL**	Следующая программа выведет строку ”is a test”: char *p; p = strchr(“this is a test”,’ ‘); cout<<p<<endl;

7 Выделение слов в строке	**char *strtok (char *str1, char *str2)** возвращает указатель на следующую лексему в строке, на которую указывает *str1*. Символы из строки, на которую указывает *str2*, используются как ограничители, определяющие лексему. Если лексема не найдена, то возвращается NULL	char *p, s[80];
		int i=1;
		cin.getline(s,80);
		//выделение слова
		p = strtok(s," \n");
		cout<<i<<" "<<p<<endl;
		while(p = strtok(NULL," \n"))
		{ i++;
		cout<<i<<" "<<p<<endl;
		}

Задание 1

1 Для заданной строки символов проверить, является ли она симметричной или нет (симметричной считается строка, которая одинаково читается слева направо и справа налево).

2 Для заданной строки символов определить сумму всех входящих в неё цифр.

3 Для заданной строки определить все входящие в неё символы. Например: строка "abccbbbabba" состоит из символов "a", "b" и "c".

4 Задана строка символов. Определить, какой символ встречается в этой

строке подряд наибольшее число раз. В ответе указать символ, образующий самую длинную последовательность, длину последовательности.

5 Для заданной строки символов, состоящей из строчных букв и пробелов, определить слово наибольшей длины, которое начинается и заканчивается на одну и ту же букву.

6 Задана строка символов, содержащая два или более слов, разделенных пробелами. Написать программу, меняющую местами все четные и нечетные слова в строке.

7 Подсчитать, сколько раз в данной строке встречается некоторая буква, вводимая с клавиатуры.

8 Из строки удалить среднюю букву, если длина строки нечетная, если четная – удалить две средние буквы.

9 Заменить все вхождения в текст некоторой буквы на другую букву (их значения вводить с клавиатуры).

10 Заменить все вхождения подстроки Str1 на подстроку Str2 (подстроки вводятся с клавиатуры)

11 Дана последовательность слов. Напечатать все слова в алфавитном порядке.

12 Дана последовательность слов. Напечатать все слова последовательности, которые встречаются в ней по одному разу.

Задание 2

Дана строка, в которой содержится осмысленное текстовое сообщение. Слова сообщения разделяются пробелами и знаками препинания.

1 Вывести только те слова сообщения, которые содержат не более чем n букв.

2 Вывести только те слова сообщения, которые начинаются с прописной буквы.

3 Вывести только те слова сообщения, которые содержат хотя бы одну цифру.

4 Удалить из сообщения все слова, которые заканчиваются на заданный символ.

5 Удалить из сообщения все слова, содержащие данный символ (без учета регистра).

6 Удалить из сообщения все однобуквенные слова (вместе с лишними пробелами).

7 Удалить из сообщения все повторяющиеся слова (без учета регистра).

8 Подсчитать, сколько раз заданное слово встречается в сообщении.

9 Подсчитать, сколько слов, состоящих только из прописных букв, содержится в сообщении.

10 Найти самое длинное слово сообщения.

11 Найти все самые длинные слова сообщения.

12 Найти самое короткое слово сообщения.

10 Лабораторная работа № 10. Работа с функциями

Цель работы: получение навыков использования пользовательских функций.

Постановка задачи: решение задач с использованием пользовательских функций.

10.1 Основные сведения

Функция – это именованная последовательность описаний и операторов, выполняющая какое-либо законченное действие.

Любая программа на C++ состоит из функций, одна из которых должна иметь имя main (с нее начинается выполнение программы). Функция начинает выполняться в момент вызова. Любая функция должна быть объявлена и определена. Объявление функции должно находиться в тексте раньше ее вызова для того, чтобы компилятор мог осуществить проверку правильности вызова.

Структура программы с использованием функций:

```
подключение библиотечных файлов(#include<>)
объявление глобальных переменных и типов
объявление функций
int main()
{
объявление локальных переменных
…
вызов функции
…
}
определение функции
```

Объявление функции (прототип, заголовок, сигнатура) задает ее имя, тип возвращаемого значения и список передаваемых параметров:

```
тип имя ([ список_параметров ]);
```

Определение функции содержит, кроме объявления, тело функции, представляющее собой последовательность операторов и описаний в фигурных скобках:

```
тип имя ([ список_параметров ])
{ тело функции:
  - объявление локальных переменных
  - операторы
  - return [выражение];
}
```

Рассмотрим составные части определения.

Тип возвращаемого функцией значения может быть любым. Если функция не должна возвращать значение, указывается тип void.

Список параметров определяет величины, которые требуется передать в функцию при ее вызове. Элементы списка параметров разделяются запятыми и для каждого параметра указывается его тип и имя. Функция также может не иметь параметров, тогда указываются просто пустые скобки ().

Оператор return служит для выхода из функции и возврата значения в точку вызова функции. Если функция описана как void, выражение не указывается. Выражение, указанное после return, преобразуется к типу возвращаемого функцией значения и передается в точку вызова.

Для **вызова** функции необходимо указать ее имя и в круглых скобках через запятую передать ей набор аргументов в соответствии с параметрами, указанными в заголовке функции:

имя (список аргументов);

Если тип возвращаемого функцией значения не void, она может входить в состав выражений или, в частном случае, располагаться в правой части оператора присваивания.

В определении, в объявлении и при вызове одной и той же функции типы и порядок следования параметров должны совпадать.

Параметры функции.

Использование параметров является основным способом обмена информацией между вызываемой и вызывающей функциями. Параметры, перечисленные в заголовке описания функции, называются *формальными*, а записанные в операторе вызова функции – *фактическими*.

При вызове функции, в первую очередь, вычисляются выражения, стоящие на месте фактических параметров; затем выделяется память под формальные параметры функции в соответствии с их типом, и каждому из них присваивается значение соответствующего фактического параметра.

Существуют два способа передачи параметров в функцию: по значению и по адресу.

Передача по значению.

Синтаксис:

– вызов функции: имя_функции (имя_фактичекого_параметра);

– определение и объявление: тип имя_функции (тип имя_формального_параметра);

При передаче по значению в стек заносятся копии значений фактических параметров, и операторы функции работают с этими копиями. Доступа к исходным значениям параметров у функции нет, и поэтому при изменении формальных параметров фактические параметры не изменяются.

Передача по адресу.

Используется два синтаксиса:

1) с помощью ссылки:

а) вызов функции:имя_функции(имя_фактического_параметра);

б) определение и объявление функции:тип имя_функции (тип &имя_формального_параметра);

2) с помощью указателя (переменная, которая хранит адрес области памяти):

а) вызов функции: имя_функции (&имя_фактичекого_параметра);

б) определение и объявление: тип имя_функции (тип *имя_ формального_параметра);

При передаче по адресу в стек заносятся копии адресов фактических пара-

метров, а функция осуществляет доступ к ячейкам памяти по этим адресам, т. е. при изменении значений формальных параметров значения фактических параметров также изменяются. При передаче по адресу в качестве фактических параметров нельзя использовать выражения, а только имена переменных:

```
#include <iostream.h>
void f(int , int* , int& );
int main()
{
int i = 1, j = 2, k= 3;
cout <<"i  j  k\n";
cout << i <<"\t"<< j <<"\t"<< k <<endl;  //1 2 3
f(i,&j,k);
//вывод результатов после вызова функции
cout << i <<"\t"<< j <<"\t"<< k <<endl; //1 3 4
return 0;
}
void f(int a, int*b, int& c)
{
a++;  (*b)++; c++;
}
```

Пример – Написать программу для вычисления

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

где n – факториал числа, $n! = 1 \cdot 2 \cdot 3 \ldots n$.

Вычисления факториала числа оформим в виде функции, в качестве параметра которой целое число, факториал которого необходимо вычислить (рисунок 10.1). Результат выполнения функции – целое число, равное факториалу параметра. В основной программе 3 раза вызываем функции, для получения факториалов чисел n, m и n – m.

Текст программы:

```
#include <iostream.h>
#include <windows.h>
#include<math.h>
char  buf[256];

// объявление функций
char* RUS(const char*);
int fact(int );

//основная функция,
//с которой начинается программа
int main()
{
  int n, m,c;
  cout<<RUS("Введите n и m");
  cin >>n>>m;
  c= fact(n)/(fact(m)*fact(n-m));
  cout <<"C="<<c; // вызов функции
  return 0;
}
```

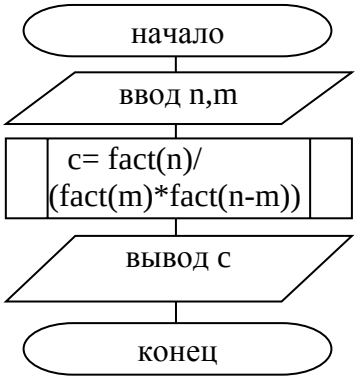
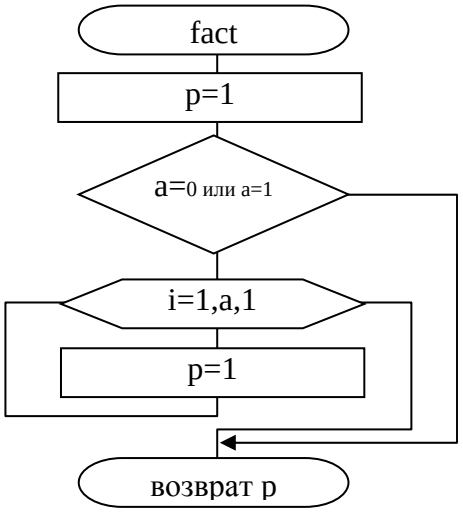


Рисунок 10.1 – Схема алгоритма программы

```
// определение функций
int fact(int a)
{
  int i,  p=1;
  if(a>1)
      for(i=1;i<=a;i++)
            p=p*i;
  return p; //возврат значения p в точку вы-
зова
}

char* RUS(const char* text)
{
   CharToOem(text, buf);
   return buf;}
```



Окончание рисунка 10.1

Задание

1 Разработать функцию min(a,b) для нахождения минимального из двух чисел. Вычислить с помощью нее минимальное значение из четырех чисел x, y, z, v.

2 Разработать функцию max(a,b) для нахождения максимального из двух чисел. Вычислить с помощью нее значение выражения z = max(x,2y – x) + + max(5x + 3y,y).

3 Разработать функцию f(n), которая для заданного натурального числа n находит значение $\sqrt{n}+n$. Вычислить с помощью нее значение выражения $\frac{\sqrt{6}+6}{2}+\frac{\sqrt{13}+13}{2}+\frac{\sqrt{21}+21}{2}$.

4 Разработать функцию f(n, x), которая для заданного натурального числа n и вещественного x находит значение выражения $\frac{x^n}{n}$. Вычислить с помощью данной функции значение выражения $\frac{x^2}{2}+\frac{x^4}{4}+\frac{x^6}{6}$.

5 Разработать функцию f(x), которая нечетное число заменяет на 0, а четное число уменьшает в 2 раза. Продемонстрировать работу данной функции на примере.

6 Разработать функцию f(x), которая число, кратное 5, уменьшает в 5 раз, а остальные числа увеличивает на 1. Продемонстрировать работу данной функции на примере.

7 Разработать функцию f(x), которая в двузначном числе меняет цифры местами, а остальные числа оставляет без изменения. Продемонстрировать работу данной функции на примере.

8 Разработать функцию f(x), которая в трехзначном числе меняет местами первую с последней цифрой, а остальные числа оставляет без изменения. Продемонстрировать работу данной функции на примере.

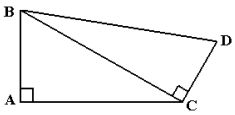


Рисунок 10.2 – Фигура ABCD

9 Разработать функцию f(a, b), которая по катетам a и b вычисляет гипотенузу. С помощью данной функции най-

ти периметр фигуры ABCD (рисунок 10.2) по заданным сторонам AB, AC и DC.

10 Разработать функцию f(x_1, y_1, x_2, y_2), которая вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и функцию d(a, b, c), которая вычисляет периметр треугольника по длинам сторон a, b, c. С помощью данных функций найти периметр треугольника, заданного координатами своих вершин.

11 Разработать функцию f(x_1, y_1, x_2, y_2), которая вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и функцию max(a, b), которая вычисляет максимальное из чисел a, b. С помощью данных функций определить, какая из трех точек на плоскости наиболее удалена от начала координат.

12 Разработать функцию f(x_1, y_1, x_2, y_2), которая вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и функцию min(a, b), которая вычисляет минимальное из чисел a, b. С помощью данных функций найти две из трех заданных точек на плоскости, расстояние между которыми минимально.

13 Разработать функцию f(x_1, y_1, x_2, y_2), которая вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и функцию t(a, b, c), которая проверяет, существует ли треугольник с длинами сторон a, b, c. С помощью данных функций проверить, можно ли построить треугольник по трем заданным точкам на плоскости.

14 Разработать функцию f(x_1, y_1, x_2, y_2), которая вычисляет длину отрезка по координатам вершин (x_1, y_1) и (x_2, y_2), и функцию t(a, b, c), которая проверяет, существует ли треугольник с длинами сторон a, b, c. С помощью данных функций проверить, сколько различных треугольников можно построить по четырем заданным точкам на плоскости.

Список литературы

1 **Колдаев, В. Д.** Основы алгоритмизации и программирования: учебное пособие / В. Д. Колдаев; под ред. проф. Л. Г. Гагариной. – Москва: Форум; ИНФРА-М, 2016. – 416 с.

2 **Дорогов, В. Г.** Основы программирования на языке С: учебное пособие / В. Г. Дорогов, Е. Г. Дорогова; под ред. проф. Л. Г. Гагариной. – Москва: Форум; ИНФРА-М, 2019. – 224 с.

3 **Немцова, Т. И.** Программирование на языке высокого уровня. Программирование на языке С++: учебное пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев; под ред. Л. Г. Гагариной. – Москва: Форум; ИНФРА-М, 2019. – 512 с.

4 **Кузин, А. В.** Программирование на языке Си / А. В. Кузин, Е. В. Чумакова. – Москва: Форум; ИНФРА-М, 2015. – 144 с.

5 **Воронцова, Е. А.** Программирование на С++ с погружением: практические задания и примеры кода / Е. А. Воронцова. – Москва: ИНФРА-М, 2016. – 80 с.

6 **Павловская, Т. А.** С/С++. Программирование на языке высокого уровня: учебник / Т. А. Павловская. – Санкт-Петербург: Питер, 2008. – 258 с.

7 **Ишкова, Э. А.** С++. Начала программирования / Э. А. Ишкова. – Москва: Бином, 2000. – 304 с.

8 **Подбельский, В. В.** Язык С++: учебное пособие / В. В. Подбельский. – Москва: Финансы и статистика, 2005. – 560 с.