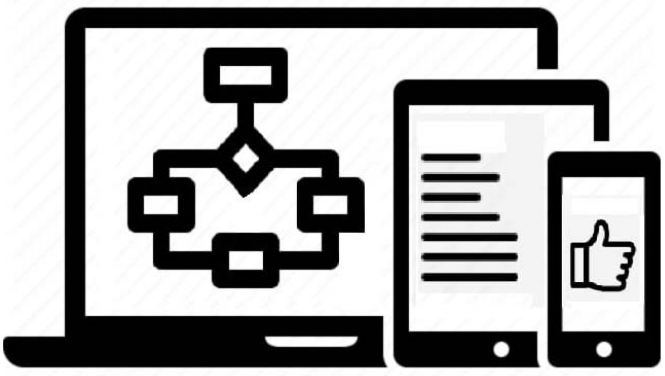
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методические рекомендации к лабораторным работам для студентов специальности 6-05-0612-03 «Системы управления информацией» дневной и заочной форм обучения

Часть 1



Могилев 2024

УДК 004.4
ББК 32.973
О75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «6» декабря 2023 г., протокол № 5

Составитель ст. преподаватель А. И. Кашпар

Рецензент канд. техн. наук, доц. В. М. Ковальчук

Методические рекомендации предназначены для выполнения лабораторных работ студентами специальности 6-05-0612-03 «Системы управления информацией» дневной и заочной форм обучения.

Учебное издание

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Часть 1

Ответственный за выпуск	В. В. Кутузов
Корректор	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 16 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, г. Могилев.

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 6-05-0612-03 «Системы управления информацией». Для студентов заочной формы обучения в осеннем семестре обязательны к выполнению лабораторные работы № 2, 4, 6, 7, 9 и 11.

Отчет по лабораторным работам оформляется индивидуально каждым студентом 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 Общие сведения

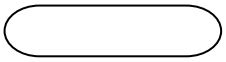
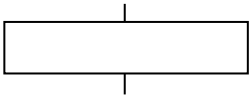
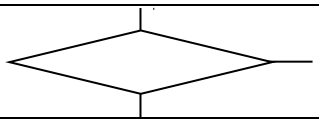
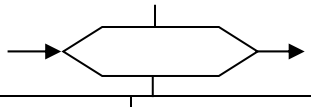
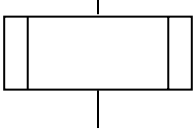
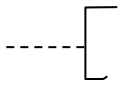
1 Алгоритмизация.

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. В каждый блок-символ входит не более одной линии потока информации. Из каждого блок-символа выходит не более одной линии потока информации.

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.

Линии потока информации, выходящие из блока «Решение», могут быть отмечены парами противоположных по смыслу знаками: условие истинно: «да», «+», «1»; условие ложно: «нет», «–», «0».

Стандартное ветвление имеет два вероятных направления (рисунок 1.1).

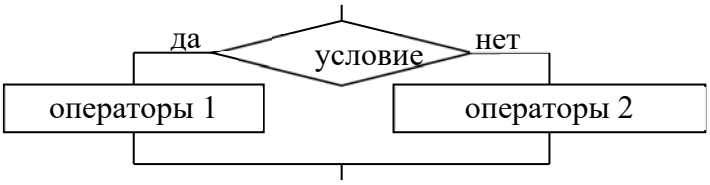


Рисунок 1.1 – Стандартное ветвление

Комбинируя проверку нескольких логических условий, можно получить многоуровневое ветвление с тремя и более вероятными направлениями.

Циклический алгоритм – фрагмент алгоритма, предусматривающая многократное повторение однотипный действий, называемых **телом цикла**.

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

Циклический алгоритм с предусловием (рисунки 1.2 и 1.4) характеризуется тем, что проверка условия расположена перед телом цикла.

Циклический алгоритм с постусловием (рисунок 1.3) характеризуется тем, что проверка условия расположена после тела цикла (лат. post – после).

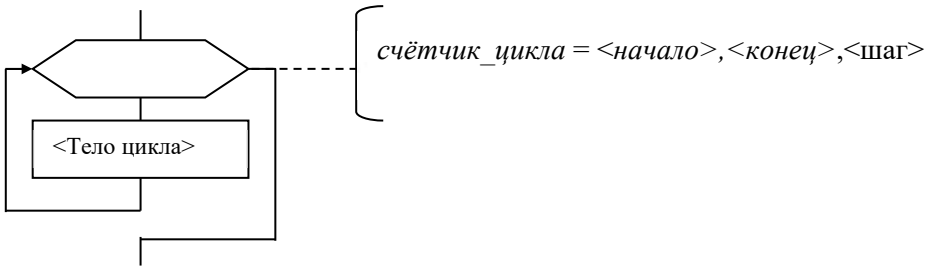


Рисунок 1.2 – Циклический алгоритм с параметром

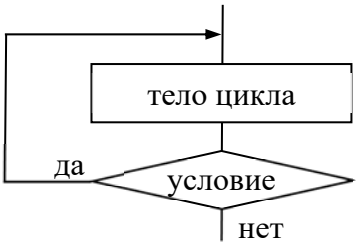


Рисунок 1.3 – Циклический алгоритм с постусловием

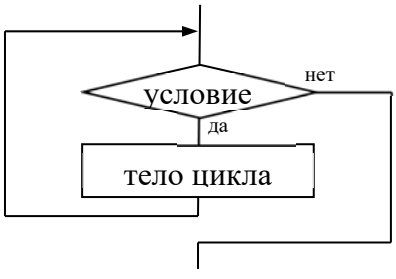


Рисунок 1.4 – Циклический алгоритм предусловием

2 Примеры алгоритмов.

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

Перед составлением алгоритма надо четко определить, *что* в нее *требуется ввести* и *что* мы должны *получить в результате*.

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 $x2$.

Блок-схема представлена на рисунке 1.5.

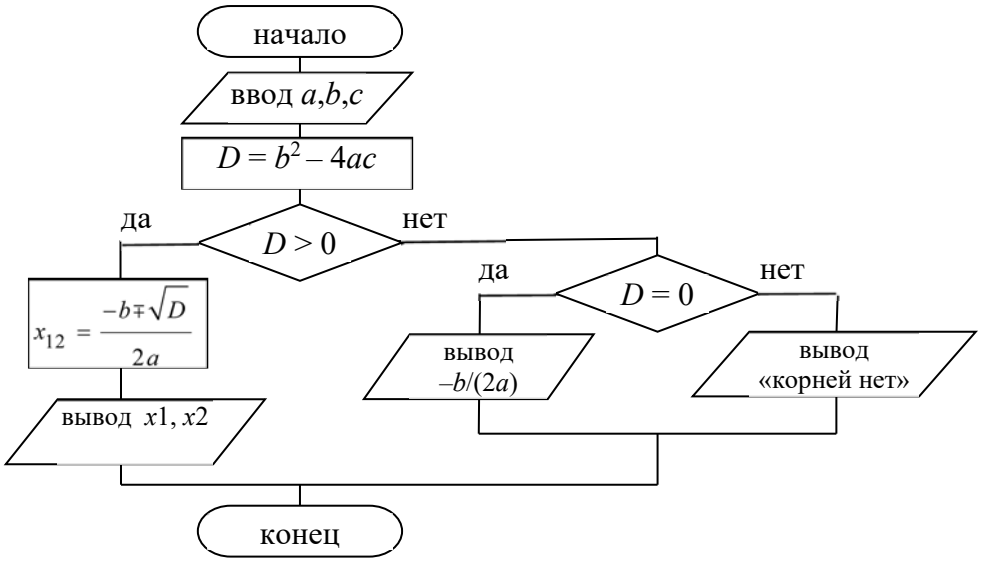


Рисунок 1.5 – Пример составления разветвляющего алгоритма

Пример 2 – Вычислить *n*-й член последовательности, заданной формулой $a_n = a_{n-1} + a_{n-2}$, если $a_1 = 1,\ a_2 = 1$.

Блок-схема решения представлена на рисунке 1.6.

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 1.2.

Задание 2

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

2 Даны три числа. Найти наименьшее из них.

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

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

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

6 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*.

7 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*.

8 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*.

9 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*.

10 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*.

11 Даны целочисленные координаты точки на плоскости. Если точка совпадает с началом координат, то вывести 0. Если точка не совпадает с началом координат, но лежит на оси OX или OY, то вывести соответственно 1 или 2. Если точка не лежит на координатных осях, то вывести 3.

12 На числовой оси расположены три точки: *A*, *B*, *C*. Определить, какая из двух последних точек (*B* или *C*) расположена ближе к *A*, и вывести эту точку и ее расстояние от точки *A*.

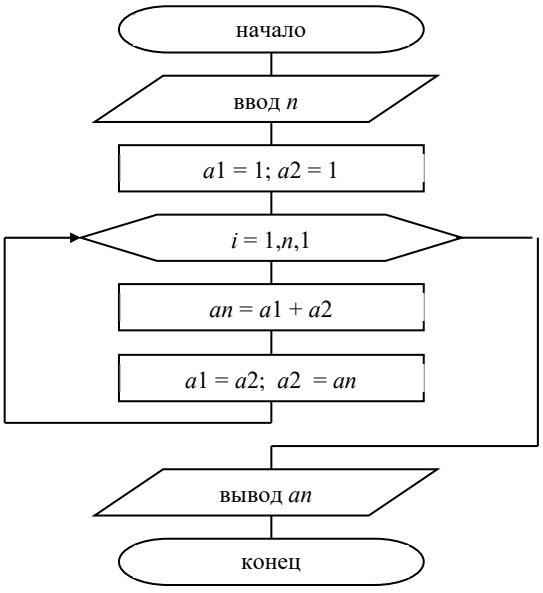


Рисунок 1.6 – Пример составления циклического алгоритма

Таблица 1.2 – Индивидуальные задания

Номер варианта	Данные 1	Данные 2	Переменная условия	Условие выполняется	Условие не выполняется
1	A_1	A_2	$B = (A_1^2 - A_2) > 0$	$C = 3{,}1B - A_1A_2^2$	$C = A_1^2A_2 - 3{,}1B$
2	X_1	X_2	$Y = (4X_1 - X_2) < 0$	$Z = 5Y^2 - X_2 + 2X_1$	$Z = X_1 + X_2 - 5Y^2$
3	E_1	E_2	$F = (3E_2 - E_2) > 0$	$Y = 4F/(E_1 - E_2)$	$Y = 4F/(E_1 - E_2)$
4	B_1	B_2	$E = (B_1 - B_2^2) < 0$	$X = 2E - B_1 + B_2^2$	$X = 2E - B_2 + B_1^2$
5	C_1	C_2	$X = (C_1C_2) > 0$	$E = 7{,}5X - C_1/C_2$	$E = C_1/C_2 - 7{,}5X$
6	D_1	D_2	$I = (D_1^2 - D_2^2) < 0$	$B = 2{,}1D_1 - 3{,}4I$	$B = 1{,}2D_2 - 3{,}4I$
7	F_1	F_2	$A = (7F_1^2 - F_2) > 0$	$D = 3{,}1A - 7F_1F_2$	$D = 7F_1F_2 - 3{,}1A$
8	H_1	H_2	$D = (H_1/H_2^2) < 0$	$F = 8D - 7{,}5H_1H_2$	$F = 4{,}5\ H_1H_2 - 6D$
9	I_1	I_2	$H = (3I_1^2 - I_2^2) > 0$	$A = H/(3{,}8I_1 - I_2)$	$A = H/(12 - 3{,}8I_1)$
10	J_1	J_2	$C = (J_1J_2 - 9) < 0$	$I = 3C - 4{,}1(J_1J_2)^2$	$I = 4\ (J_1J_2)^2 - 3C$
11	K_1	K_2	$J = (K_1 - 5K_2^2) > 0$	$H = 2{,}5J - K_1K_2$	$H = K_1K_2 - 2{,}5J$
12	L_1	L_2	$K = (L_1^2/L_2) < 0$	$J = 9K - 3L_1 + L_2$	$J = 3L_1 - L_2 - 9K$

Задание 3

Составить блок-схему алгоритма решения циклической задачи вычисления значения функции в зависимости от значения переменной аргумента. Значения переменной аргумента должны изменяться от начального до конечного значения с заданным шагом изменения. Исходные данные к заданию находятся в таблице 1.3.

Таблица 1.3 – Индивидуальные задания

Номер варианта	Начальное значение	Конечное значение	Значение шага	Выражение для расчета значений функции
1	*Xn*	*Xk*	*Hx*	$Y = 2{,}23X^2 + 3{,}12X - 1{,}95$
2	*Yn*	*Yk*	*Hy*	$Z = -1{,}25Y^2 + 0{,}46Y - 0{,}88$
3	*Zn*	*Zk*	*Hz*	$A = 2{,}85Z^2 - 4{,}15Z + 6{,}05$
4	*An*	*Ak*	*Ha*	$B = -1{,}45A^2 + 9{,}15A + 12{,}5$
5	*Bn*	*Bk*	*Hb*	$C = 0{,}52B^2 + 2{,}52B + 7{,}84$
6	*Cn*	*Ck*	*Hc*	$D = -0{,}15C^2 - 5{,}33C - 9{,}21$
7	*Dn*	*Dk*	*Hd*	$E = 1{,}08D^2 - 7{,}22D - 2{,}43$
8	*En*	*Ek*	*He*	$F = 4{,}02E^2 - 8{,}49E + 3{,}14$
9	*Fn*	*Fk*	*Hf*	$G = -2{,}36P^2 + 6{,}07F + 8{,}3$
10	*Gn*	*Gk*	*Hg*	$I = -3{,}44G^2 - 4{,}72G + 5{,}57$
11	*In*	*Ik*	*Hi*	$J = 0{,}97I^2 - 1{,}75I + 4{,}32$
12	*Jn*	*Jk*	*Hj*	$K = 1{,}06J^2 + 5{,}67J - 6{,}98$

Задание 4

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

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

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

4 Дано вещественное число A и целое число N (> 0). Найти сумму $1 + A + A^2 + ... + A^N$.

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

6 Дано целое число $N > 0$. Найти значение выражения $1.1 - 1.2 + 1.3 - ...$ (N слагаемых, знаки чередуются).

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

8 Дано вещественное число X ($|X| < 1$) и целое число $N > 0$. Найти значение выражения $X - X^3/3 + X^5/5 - ... + (-1)^N \cdot X^{2N+1}/(2N + 1)$.

9 Дано целое число $N > 0$. Найти сумму $1! + 2! + 3! + ...+ N!$ ($N! = 1 \cdot 2 \cdot ... \cdot N$). Чтобы избежать целочисленного переполнения, проводить вычисления с помощью вещественных переменных и вывести результат как вещественное число.

10 Дано целое число $N > 0$. Найти сумму $1 + 1/(1!) + 1/(2!) + 1/(3!) + ... + 1/(N!)$ ($N! = 1 \cdot 2 \cdot ... \cdot N$). Полученное число является приближенным значением константы e.

11 Дано вещественное число X ($|X| < 1$) и целое число $N > 0$. Найти значение выражения $X - X^2/2 + X^3/3 - ... + (-1)^{N-1} \cdot X^N/N$.

12 Дано целое число N (> 1) и две вещественные точки на числовой оси: A, B ($A < B$). Отрезок [A, B] разбит на N равных отрезков. Вывести H – длину каждого отрезка, а также набор точек A, $A + H$, $A + 2H$, ..., B, образующих разбиение отрезка.

2 Лабораторная работа № 2. Работа с главным меню системы Visual Studio. Программирование линейных алгоритмов

Цель работы:

- изучение интегрированной среды;
- освоение простейшей структуры программы на языках C#/C++;
- получение навыков в организации ввода-вывода на языках C#/C++;
- изучение базовых типов данных языков C#/C++.

Постановка задачи. Напишите программу для расчета по двум формулам.

2.1 Общие сведения

1 Линейная программа.

Если в программе все операторы выполняются последовательно, один за другим, такая программа называется *линейной.* Рассмотрим в качестве примеров программу, вычисляющую результат по заданной формуле.

Задача. Расчет по формуле. Написать программу, которая переводит температуру в градусах по Фаренгейту в градусы Цельсия по формуле

$$C = 5/9 \, (F - 32),$$

где C – температура по Цельсию;

F – температура по Фаренгейту.

Перед написанием любой программы надо четко определить, что в нее требуется ввести и что мы должны получить в результате.

В данном случае:

– исходные данные – одно вещественное число (температура по Цельсию);

– результат – другое вещественное число (температура по Фаренгейту).

Перед написанием программы откроем интегрированную среду Visual Studio: Пуск/Программы/Microsoft Visual Studio … .

Далее создадим проект. Для этого:

– выберем пункт «Создание проекта»;

– в открывшемся окне «Создание проекта» выберем ***Консольное приложение*** (.NetFramework) C# или ***Консольное приложение*** C++; нажимаем кнопку «Далее»;

– в открывшемся окне «Настроить новый проект» введите имя проекта в текстовом поле (например, Lab2) и определите положение на диске, куда нужно сохранять ваш проект (например, Z:\Gr\FIO) и подтвердите кнопкой «Создать».

В результате появится шаблон консольного приложения, в котором присутствует только заготовка текста программы (рисунок 2.1).

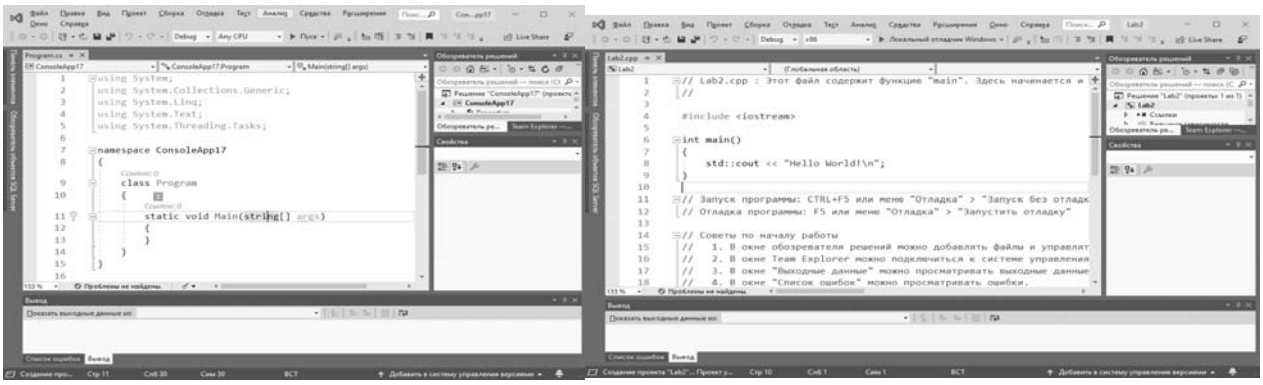


Рисунок 2.1 – Шаблоны консольного приложения

Основное пространство экрана занимает окно редактора, в котором располагается текст программы, созданный средой автоматически. Текст представляет собой каркас, в который программист будет добавлять нужный код. При этом зарезервированные слова отображаются синим цветом, комментарии – зеленым, основной текст – черным.

Теперь рассмотрим сам текст программы.

Using System – это директива, которая разрешает использовать имена стандартных классов из пространства имен **System**. Ключевое слово **namespace** создает для проекта свое собственное пространство имен, которое по умолчанию называется именем проекта.

Автоматически создан класс с именем **Program**. Данный класс содержит только один метод – метод **Main()**. Метод Main() является точкой входа в программу, т. е. именно с данного метода начнется выполнение приложения. Каждая программа на языках C#/C++ должна иметь метод Main()/main().

Добавим в метод Main() следующий код на C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Lab2
{
    class Program
    {
        static void Main(string[] args)
        {
            double fahr, cels;   //1
            Console.WriteLine(" Введите температуру По Фаренгейту"); //2
            fahr=Convert.ToDouble(Console.ReadLine()); //3
            cels=5/9 * (fahr - 32);  //4
            Console.WriteLine(" По Фаренгейту: "+fahr+", в градусах Цельсия: "+cels);  //5
            Console.ReadKey();  //6
        }
    }
}
```

Рассмотрим каждую строку программы отдельно.

В методе Main() в фигурных скобках { } записывается тело метода, т. е. те операторы, которые требуется выполнить.

Любая заготовка при написании программы имеет вид:

```
static void Main(…)
{
    объявление переменных;
    ввод исходных данных;
    расчет результата;
    вывод результата;
}
```

Для хранения исходных данных и результатов надо выделить место в оперативной памяти. Для этого служит оператор 1. В программе требуется хранить два значения: температуру по Цельсию *cels* и температуру по Фаренгейту *fahr*, поэтому в операторе определяются две переменные. Имена переменным дает программист, исходя из их назначения. **Имя может состоять только из букв, цифр и знака подчеркивания и не должно начинаться с цифры.** При описании любой переменной нужно указать ее **тип**. Поскольку температура может принимать не только целые значения, для переменных выбран вещественный тип **double**.

Основные типы:

float , **double** – вещественные;

int – целочисленные;

char – символьный;

string – строковый;

bool – логический.

Для того чтобы пользователь программы знал, в какой момент требуется ввести с клавиатуры данные, применяется так называемое приглашение к вводу (оператор 2). Здесь **Console** – имя стандартного класса из пространства имен System для работы с консольными приложениями. Его метод **WriteLine** выводит на экран текст, заданный в кавычках.

В операторе 3 выполняется ввод с клавиатуры одного числа в переменную *fahr* с помощью метода **ReadLine()**. Данный метод возвращает введенную строку, а с помощью метода Convert – преобразует данные в необходимый тип (например, Convert.ToInt32() – к целому типу, Convert.ToDouble() – к вещественному).

В операторе 4 вычисляется выражение, записанное справа от операции присваивания (=), и результат присваивается переменной *cels*. Сначала **целая** константа 5 будет поделена на **целую** константу 9, затем результат этой операции умножен на (*fahr* – 32).

Для вывода результата в операторе 5 применяется метод WriteLine(). Выводится цепочка, состоящая из четырех элементов. Это строка *«По Фаренгейту:»*, значение переменной *fahr*, строка *«, в градусах Цельсия:»* и значение переменной *cels*. Знак «+» складывает два значение. Если это числа – складывает числа, а если строки – то соединяет их вместе.

Последний оператор 6 **ReadKey()** задает задержку для консоли, чтобы окно сразу же не закрывалось, а ожидало нажатия любой кнопки.

Для запуска программы следует нажать клавишу F5 или выполнить команду **Отладка (Debug) – Начать отладку (Start Debugging) или кнопку**  .

При запуске программы возникает проблема: результат выполнения программы всегда равен нулю при вводе данных. Причина: константы 5 и 9 имеют целый тип, что приводит к целочисленному делению и результату равному нулю. Исправление: Записать хотя бы одну из констант в виде вещественного числа: cels = 5.0 / 9 * (fahr – 32).

По аналогии создаем программу на С++ для вычисления значение функции

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

Перед написанием любой программы надо четко определить, что в нее требуется ввести и что мы должны получить 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 на С++:

```
//директивы препроцессора
#include <iostream>    //для использования cin, cout
#include <math.h>         // для использования fabs(), pow(), sin()
using namespace std;   //чтобы каждый раз не писать std::cout, std::cin
int main()
{system(«chcp 1251»);//функция для вывода кириллицы в консоли
 float x, y, f;   //объявление переменных вещественного типа
 cout <<»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 << «Результат: f(x,y)= « << f << endl;   //вывод результата
 return 0; }
```

Синтаксис программы на С++ очень схож с синтаксисом С#, поэтому рассмотрим различия. Для вывода данных на экран (в консольное окно) используется объект std::cout (который находится в библиотеке iostream). Для вывода нескольких предложений на одной строке оператор вывода << нужно использовать несколько раз. Если текст нужно вывести раздельно (на нескольких строках) – используйте std::endl.

Std::cin получает данные от пользователя с помощью оператора ввода >>.

Директива using позволяет использовать все имена без имени пространства имен в namespace в качестве явного квалификатора, поэтому, добавив команду using namespace std; в программе используют cout, cin, endl без std:: .

В языке С# предоставляется целый класс математических методов – **Math** (таблица 2.1). Для вызова метода необходимо прописать: Math.Имя_метода(), например, x^2 записывается в виде: y=Math.Pow(x,2).

Обратите внимание на то, что вычисление синуса, косинуса и т. д. вычисляется в радианах. Поэтому если вам нужны градусы, нужно конвертировать `double radian = gradus * Math.PI / 180;`

Таблица 2.1 – Математические методы С#/С++

Метод С#	Функция С++	Назначение
Math.Abs	abs/fabs	Возвращаем модуль числа/вещественного числа
Math.Acos	acos	Арккосинус. Определяется угол (в радианах), косинус которого равен указанному числу

Окончание таблицы 2.1

Метод C#	Функция C++	Назначение
Math.Asin	asin	Арксинус. Также определяет угол
Math.Atan	atan	Арктангенс
Math.Cos	cos	Возвращает косинус угла, заданного в радианах
Math.Exp	exp	Экспонента
Math.Log	log	Вычисление логарифма, Например, Math.Log(x,2) задает $\log_2 x$
Math.Pow	pow	Вычисляет число, возведенное в степень: a^x
Math.Sin	sin	Возвращает синус угла
Math.Sqrt	sqrt	Возвращает квадратный корень
Math.Tan	tan	Возвращает тангенс угла

Задание

Напишите программу для расчета по двум формулам. 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) $z_1 = \frac{\sin\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$; $z_2 = \mathrm{tg}3\alpha$;

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

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) $z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$; $z_2 = \mathrm{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$;

11) $z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$;
$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$;

12) $z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \frac{\cos 2\alpha}{1 + \cos 2\alpha}$;
$z_2 = \mathrm{ctg}\left(\frac{3}{2}\pi - \alpha\right)$.

3 Лабораторная работа № 3. Организация ввода/вывода в консольных программах

Цель работы:

– освоение простейшей структуры программы на языках C#/C++;

– получение навыков в организации форматированного ввода-вывода.

Постановка задачи. Написать программы, которые реализует диалог с пользователем. Вывод производить двумя способами: слияние строк; форматированный вывод.

3.1 Общие сведения

В **C#** для работы с консолью используется стандартный класс Console, определенный в пространстве имен System.

Вывод данных. Метод WriteLine позволяет организовывать вывод данных на экран. Существует несколько способов применения данного метода:

```
1  Console.WriteLine(x);     //на экран выводится значение идентификатора x
2  Console.WriteLine("x=" + x +"y=" + y); /* на экран выводится строка, образованная слиянием строки  "x=", значения x, строки "y=" и значения y */
3  Console.WriteLine("x={0} y={1}", x, y);/* на экран выводится строка, формат которой задан первым аргументом метода, при этом вместо параметра {0} выводится значение x, а вместо {1} – значение y*/
4  Console.WriteLine($"x={x} y={y}");     /* упрощённый аналог 3*/
```

Управление форматом числовых данных. Первым аргументом WriteLine указывается строка вида {*n,m:<спецификатор>k*} – где *n* определяет номер идентификатора из списка аргументов метода WriteLine; *<спецификатор>* – определяет формат данных; *m* – количество позиций (размер поля вывода), отводимых под значение данного идентификатора (при этом значение идентификатора выравнивается по правому краю); *k* – количество позиций для дробной части значения идентификатора. В качестве спецификаторов могут использоваться значения, представленные в таблице 3.1.

Таблица 3.1 – Форматы выводы

Параметр	Формат
C или c	Денежный. По умолчанию добавляет валюту «р.», *k* – задает количество десятичных разрядов
D или d	Целочисленный (используется только с целыми числами), *k* – задает минимальное количество цифр (результат дополняется начальными нулями)
E или e	Экспоненциальное представление чисел, *k* – задает количество символов после запятой (по умолчанию 6)
F или f	Представление чисел с фиксированной точкой *k* – задает количество символов после запятой
G или g	Общий формат (или экспоненциальный, или с фиксированной точкой), *k* – задает количество символов после запятой
N или n	Стандартное форматирование с использованием запятых и пробелов в качестве разделителей между разрядами, *k* – задает количество символов после запятой. По умолчанию – 2, если число целое, то ставятся нули
X или x	Шестнадцатеричный формат
P или p	Процентный

Например, в результате выполнения фрагмента кода

```
Console.WriteLine("C Format:{0,14:C} \t{0:C2}", 12345.678);
Console.WriteLine("D Format:{0,14:D} \t{0:D6}", 123);
Console.WriteLine("E Format:{0,14:E} \t{0:E8}", 12345.6789);
Console.WriteLine("G Format:{0,14:G} \t{0:G10}", 12345.6789);
Console.WriteLine("N Format:{0,14:N} \t{0:N4}", 12345.6789);
Console.WriteLine("X Format:{0,14:X} ", 1234);
Console.WriteLine("P Format:{0,14:P} ", 0.9);
```

на экране получим результат:

```
C Format:   12 345,68р.          12 345,68р.
D Format:            123         000123
E Format: 1,234568E+004          1,23456789E+004
G Format:    12345,6789          12345,6789
N Format:     12 345,68          12 345,6789
X Format:           4D2
P Format:        90,00%
Для продолжения нажмите любую клавишу . . .
```

Ввод данных. Для ввода данных обычно используется метод ReadLine, реализованный в классе Console. Особенностью данного метода является то, что в качестве результата он возвращает строку (string). Например,

```
string s = Console.ReadLine();
Console.WriteLine(s);
```

Для того чтобы получить числовое значение необходимо воспользоваться преобразованием данных. Например,

```
int x = int.Parse(Console.ReadLine()); //преобразование введенной строки в число
Console.WriteLine(x);
```

Для преобразования строкового представления целого числа в тип int мы используем метод int.Parse(), который реализован для всех числовых типов данных. Таким образом, если нам потребуется преобразовать строковое представление в вещественное, используется float.Parse() или double.Parse().

В C++, кроме рассмотренных ранее операторов ввода-вывода cin, cout, можно использовать функции форматированного ввода-вывода данных scanf, printf (прототипы содержатся в файле stdio.h).

Функция printf() обеспечивает форматированный вывод. Ее можно записать в следующем формальном виде:

```
printf ("управляющая строка", аргумент_1, аргумент_2,...);
```

Управляющая строка содержит компоненты трех типов: обычные символы, которые просто выводятся на экран дисплея; спецификации преобразования, каждая из которых вызывает вывод на экран очередного аргумента из последующего списка; управляющие символьные константы.

Каждая спецификация преобразования начинается со знака «%» и заканчивается некоторым символом, задающим преобразование (таблица 3.1). Между знаком «%» и символом преобразования могут встречаться другие знаки в соответствии со следующим форматом:

```
% [признаки] [ширина_поля] [.точность] [F|N|h|l|L] c_n
```

Все параметры в квадратных скобках не являются обязательными.

Значения c_n (символ преобразования) указаны в таблице 3.1; признак

минус (–) указывает, что преобразованный параметр должен быть выровнен влево в своем поле, признак плюс (+) требует вывода результата со знаком; ширина_поля – число, задающее минимальный размер поля; .точность – число, количество цифр, выводимых справа от десятичной точки в значениях типов float или double.

Например, результат вызова функции:

```
printf("d Format:%3d \nf format: %10.3f ", 5, 1.1);
```

```
d Format:  5
f format:      1.100
```

.

Функция scanf( ) обеспечивает форматированный ввод. Ее можно записать в следующем формальном виде:

```
scanf("управляющая строка", аргумент_1, аргумент_2,...);
```

Аргументы scanf( ) должны быть указателями на соответствующие значения. Для этого перед именем переменной записывается символ &.

Управляющая строка содержит спецификации преобразования и используется для установления количества и типов аргументов. Например, ввести целое число (int a;), символ (char b;) и вещественное число (float t;) можно так:

```
scanf("%d", &a);
scanf("%c%f", &b, &t);
```

Задание 1

Написать программу, которая, реализует диалог с пользователем (таблица 3.2).

Таблица 3.2 – Варианты заданий

Задание	Результат
1 Запрашивает с клавиатуры три целых числа и выводит на экран сумму данных чисел	a = 24 b = 5 c = 36 24 + 5 + 36 = 65
2 Запрашивает с клавиатуры два целых числа, и выводит на экран сумму данных чисел	a = 23 b = 43 23 + 43 = 66
3 Запрашивает с клавиатуры сумму вклада и процент по вкладу и выводит на экран следующее сообщение (вклад без капитализации – все начисления в конце года)	Введите сумму вклада = 12345,67 Введите процент по вкладу = 14,5 Через год сумма на вкладе = 14 135,79 р.
4 Запрашивает с клавиатуры два вещественных числа и выводит на экран произведение данных чисел (с точностью до одного знака после запятой)	a = 3,45 b = 12,1 3,5 * 12,1 = 41,7
5 Запрашивает с клавиатуры три вещественных числа и выводит на следующее сообщение (с точностью до двух знаков после запятой)	a = 1,4 b = 2,567 c = 10 (1,40+2,57)+10,00=1,40+(2,57+10,00)
6 Запрашивает с клавиатуры имя человека и его возраст и выводит на экран следующее сообщение	Как тебя зовут? Вася Сколько тебе лет? 15 Вася, ты родился в 2009 году
7 Запрашивает с клавиатуры два целых числа и выводит на экран сумму данных чисел в прямом и обратном порядке	a = 12 b = 34 12 + 34 = 34 + 12

Окончание таблицы 3.2

Задание	Результат
8 Запрашивает с клавиатуры номинал купюры и количество купюр, и выводит экран следующее сообщение	Номинал купюры: 100 Количество купюр: 23 Сумма денег: 2 300,00 р.
9 Запрашивает с клавиатуры два вещественных числа и выводит на экран результат деления первого числа на второе (с точностью три знака после запятой)	a = 345,6 b = 12,345 345,600 / 12,345 = 27,995
10 Запрашивает с клавиатуры сумму вклада и процент по вкладу и выводит на экран следующее сообщение (вклад без капитализации – все начисления в конце года)	Введите сумму вклада: 12345,67 Введите процент по вкладу: 14,5 Через год сумма на вкладе: 1 790,12 р.

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 расстояние между двумя точками с заданными координатами (x_1, y_1) и (x_2, y_2).

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 Дана площадь круга. Найти длину окружности, ограничивающей этот круг. В качестве значения *Pi* использовать 3,14.

8 Ввести с клавиатуры координаты трех вершин треугольника (x_1, y_1), (x_2, y_2), (x_3, y_3). 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 V_1 км/ч, второго – V_2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга.

4 Лабораторная работа № 4. Программирование разветвляющихся алгоритмов

Цель работы:

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

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

4.1 Общие сведения

Оператор if. Большинство операторов управления программой основываются на проверке условий, определяющих, какого рода действие необходимо выполнить. Стандартная форма записи оператора **if** следующая:

```
if (условие)
     оператор1;
else
     оператор2;
```

где *оператор* может быть простым или составным. Надо помнить, что в C#/C++ *составной оператор* (*блок*) – это группа операторов, заключенных в фигурные скобки. Оператор **else** не обязателен. Если *условие* 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 оба кода одновременно.

В условии могут использоваться следующие операции:

Операции отношений (сравнения):

<	– меньше, чем;	>=	– больше или равно, чем;
>	– больше, чем;	==	– равно;
<=	– меньше или равно, чем;	!=	– не равно.

Логические операции:

&& – конъюнкция (И) арифметических операндов или отношений;
|| – дизъюнкция (ИЛИ) арифметических операндов или отношений.

Результат операций логический: false (ложь) или true (истина).

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).

Лесенка if-else-if. Синтаксис:

```
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**.

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 1 (***C#***) – Написать программу для вычисления значения функции

$$F = \begin{cases} ax^2 + b & \text{при } x < 0 \text{ и } b \neq 0; \\ \dfrac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0; \\ \dfrac{1}{3}xc & \text{в остальных случаях,} \end{cases}$$

где a, b, c – целые числа;

x – действительное число.

В данном примере:

– исходные данные – три целых числа a, b и c и вещественное число x;

– результат – вещественное число F.

```
static void Main(string[] args)
{   int a, b, c;
    double x, F;
    Console.WriteLine("Введите a, b, c(целые) и x(вещественное)");
    a = int.Parse(Console.ReadLine());
    b = int.Parse(Console.ReadLine());
    c = int.Parse(Console.ReadLine());
    x = double.Parse(Console.ReadLine());
    if (x < 0 && b != 0)
        F = a * Math.Pow(x, 2) + b;
    else if (x > 0 && b == 0)
        F = (x - a) / (x - c);
    else
        F = 1.0 / 3 * x * c;
    Console.WriteLine("При a={0:d}, b={1:d}, c={2:d} и x={3:f2} - результат
F={4:f3}", a, b, c, x, F);
    Console.ReadKey();        }
```

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

В данном примере:

– исходные данные – три вещественных числа a, b и c (коэффициенты);

– результат – корни квадратного уравнения, вещественные числа $x1$ и $x2$.

```
#include <iostream>    //для использования cin, cout
#include <math.h>         // для использования sqrt(), pow()
```

```
using namespace std;
int main()
{system("chcp 1251");
 float a, b, c, x1, x2, D;   //объявление переменных вещественного типа
 cout << "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 дискременанта
 if (D > 0)
 {
     x1 = (-b - sqrt(D)) / (2 * a);
     x2 = (-b + sqrt(D)) / (2 * a);
     //вывод результата
     cout << "Уравнение имеет два корня x1=" << x1 << ", x2= " << x2 << endl;
 }
 else if (D == 0)
     cout << "Уравнение имеет корень x1= " << -b / (2 * a) << endl; //вывод
 else
     cout << "Уравнение не имеет корней" << endl; //вывод результата
 return 0;}
```

Задание 1

Написать программу для вычисления значения функции в зависимости от выполнения следующих условий.

1 $$y=\begin{cases}\sin x & \text{при } x\le 0;\\ \operatorname{arctg} x & \text{при } 0<x\le \pi/4;\\ \log_2 x & \text{при } \pi/4<x\le 32;\\ \dfrac{1}{x+2} & \text{в остальных случаях.}\end{cases}$$

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

3 $$z=\begin{cases}\dfrac{1}{x}+\dfrac{1}{y} \text{ при } x<-10 \text{ и } y<-5;\\ \dfrac{x-y}{x+y} \text{ при } -10\le x<0 \text{ и } -5\le y<0;\\ \dfrac{\sin x}{\cos y} \text{ при } 0\le x<2\pi \text{ и } 0\le y<\pi/2;\\ \ln\left(x^2+y^2\right) \text{ в остальных случаях.}\end{cases}$$

4 $$z=\begin{cases}\operatorname{arctg}\dfrac{x}{y} & \text{при } y\ne 0 \text{ и } |x|>|y|;\\ \arcsin\dfrac{x}{y} & \text{при } y\ne 0 \text{ и } |x|\le|y|;\\ 0 & \text{в остальных случаях.}\end{cases}$$

5 $$y=\begin{cases}0 & \text{при } x<0;\\ \dfrac{1}{x^2+1} & \text{при } 0\le x\le 1;\\ x^3 & \text{при } 1<x\le 4;\\ 62+\log_8 x & \text{при } x>4.\end{cases}$$

6 $$y=\begin{cases}0 & \text{при } x\le 0;\\ 1/x & \text{при } 0<x\le 1;\\ x^2 & \text{при } 1<x\le 4;\\ 14+\log_2 x & \text{в остальных случаях.}\end{cases}$$

7 $$z=\begin{cases}\dfrac{2}{x}-\dfrac{4}{y} \text{ при } x<-20 \text{ и } y<-10;\\ \dfrac{x-y-2}{x+y} \text{ при } -20\le x<0 \text{ и } -10\le y<0;\\ \dfrac{\sin x+\cos x}{\cos y} \text{ при } 0\le x<2\pi \text{ и } 0\le y<\pi/2;\\ \log_4\left(x^2+y^2\right) \text{ в остальных случаях.}\end{cases}$$

8 $$z=\begin{cases}\sin\dfrac{x+y}{y}, & \text{при } y\ne 0 \text{ и } |x|>|y|;\\ \arccos\dfrac{x}{y}, & \text{при } y\ne 0 \text{ и } |x|\le|y|;\\ \pi, & \text{в остальных случаях.}\end{cases}$$

9 $z=\begin{cases} x+y & \text{при } x<0 \text{ и } y\le 0; \\ \operatorname{arctg}\dfrac{x}{y} & \text{при } 0\le x<10 \text{ и } 0<y\le 8; \\ 0 & \text{в остальных случаях.} \end{cases}$

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

Задание 2

Варианты заданий взять из лабораторной работы № 1 (задание 2).

5 Лабораторная работа № 5. Программирование с использованием оператора switch

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

Постановка задачи. Напишите программу для решения задач.

5.1 Основные сведения

Оператор выбора switch. Оператор выбора *switch* предназначен для разветвления процесса вычислений по нескольким направлениям:

```
switch ( <выражение> )
{
  case <константное_выражение_1>:  [<оператор 1>]; <оператор перехода>;
  case <константное_выражение_2>:  [<оператор 2>]; <оператор перехода>;
   ...
  case <константное_выражение_n>:  [<оператор n>]; <оператор перехода>;
  [default: <оператор>; ]
}
```

Выражение, стоящее за ключевым словом *switch*, должно иметь арифметический, символьный, строковый (только для C#) тип или тип указатель. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящим после *switch* или приводиться к нему. В таблице 5.1 приведены примеры использования оператора *switch.*

Задание 1

1 Определить, является ли введенная буква русского алфавита гласной.

2 Написать программу преобразования цифр в слова.

3 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.

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

6 Написать алгоритм, позволяющий получить словесное наименование школьных оценок.

7 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.

8 В зависимости от того введена ли открытая скобка или закрытая, напечатать «открытая круглая скобка» или «закрытая фигурная скобка». (Учитывать круглые, квадратные, фигурные скобки).

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

10 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.

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

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

Таблица 5.1 – Примеры использования оператора switch

Пример 1 (C#)	Пример 2 (C++)

Дан порядковый номер дня недели, вывести на экран его название.
`static void Main()`
`{ Console.Write("n=");`
`byte n = byte.Parse(Console.ReadLine());`
`switch (n)`
`{case 1: Console.WriteLine("ПН"); break;`
`case 2: Console.WriteLine("ВТ"); break;`
`case 3: Console.WriteLine("СР"); break;`
`case 4: Console.WriteLine("ЧТ"); break;`
`case 5: Console.WriteLine("ПТ"); break;`
`case 6: Console.WriteLine("СБ"); break;`
`case 7: Console.WriteLine("ВС"); break;`
`default: Console.WriteLine("Ошибка");`
`break;`
`}`
`Console.ReadKey();`
`}`
	По номеру месяца вывести название поры года.
	`#include <iostream>`
	`using namespace std;`
	`int main()`
	`{ system("chcp 1251");`
	`int M;`
	`cout<<"Введите номер месяца: ";  cin>>M;`
	`switch (M)`
	`{ case 1: case 2: case 12:`
	`cout << "ЗИМА" << endl; break;`
	`case 3: case 4: case 5:`
	`cout << "ВЕСНА" << endl; break;`
	`case 6: case 7: case 8:`
	`cout << "ЛЕТО" << endl; break;`
	`case 9: case 10: case 11:`
	`cout << "ОСЕНЬ" << endl; break;`
	`default: cout << "Ошибка" << endl; break;`
	`}  return 0;}`

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

6 Лабораторная работа № 6. Программирование циклических алгоритмов

Цель работы:
– изучение циклических операторов языков C#/C++;
– получение навыков в программирование цикла с параметром for.

Постановка задачи. Разработать программы циклических процессов:
1) вывести таблицу значений функции на заданном отрезке;
2) вычисление значений конечной суммы или произведения.

6.1 Общие сведения

В языках программирования циклы позволяют выполнять набор операторов, пока выполняется некоторое условие.

Оператор for. Операторы цикла используются для организации многократно повторяющихся вычислений. Любой цикл состоит из тела цикла, т. е. 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 C# выполняют тело цикла, пока условие истинно.

Стандартный вид цикла **for** следующий:

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

Оператор **for** имеет три главные части. *Инициализация* – это выражение присваивания, используемое для установки начального значения переменной цикла. *Условие* – это выражение, определяющее условие работы цикла. *Модификация* – это 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 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 6.1).

В данной программе переменная **x** изначально установлена в 1. Поскольку **x** меньше 4, выводится с помощью WriteLine() значение 1, после чего **x** увеличивается 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.

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.

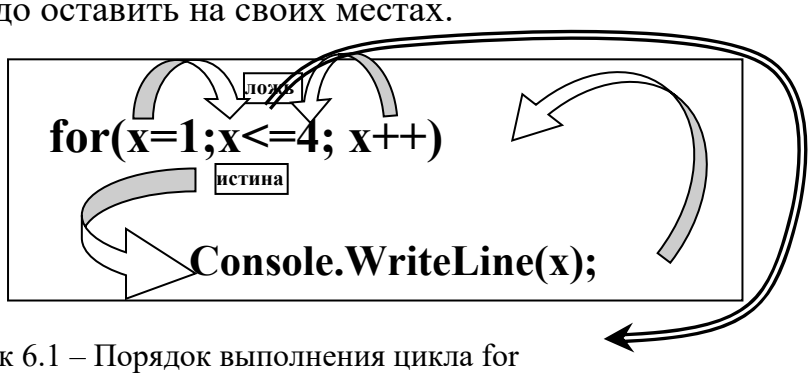


Рисунок 6.1 – Порядок выполнения цикла for

Ниже приведен ***пример*** цикла for, повторяющего сразу несколько операторов:

```
for (x=100; x >=65; x–=5)
{  z = Math.Sqrt(x);
   Console.WriteLine(“x=” + x + x “ z= “ + z);
}
```

Как Sqrt(), так и *WriteLine* (), вызываются и выполняются, пока **x** не меньше 65. В цикле переменная **x** *уменьшается* на 5 от 100 до 65 включительно.

Сокращенные операции в C#:

++ – увеличение на единицу; например *a*++ то же самое что и *a*=*a*+1;

-- – уменьшение на 1(*a*--).

f + = 5 – увеличение на 5, то же самое что и *f* = *f* + 5;

*x** = 10 – увеличение в 10 раз, то же самое что и *x*= *x**10.

***Пример 1* (*C#*)** – Построить таблицу значений для функции

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

Значения *a*, *b*, $X_{нач}$, $X_{кон}$, *dX* ввести с клавиатуры.

Исходными данными являются начальное значение аргумента *Xn*, конечное значение *Xk*, шаг изменения аргумента *dX* и параметры *a, b, c*. Все величины – вещественные. Программа должна выводить таблицу из двух столбцов – значений аргумента *x* и соответствующих им значений функции *f*.

Текст программы:

```
static void Main(string[] args)
{   try  { //объявление переменных вещественного типа
    double a, b, Xn, Xk, dX, x, f;
    //ввод исходных данных
    Console.WriteLine("Введите коэффициенты a и b:");
    a = double.Parse(Console.ReadLine());
    b = double.Parse(Console.ReadLine());
    Console.WriteLine("Введите Хнач, Хкон и шаг dX:");
```

```
    Xn = double.Parse(Console.ReadLine());
    Xk = double.Parse(Console.ReadLine());
    dX = double.Parse(Console.ReadLine());
    for (x = Xn; x <= Xk; x += dX)  //условия цикла
    {   if (x <= 0 && a == b)
            f = Math.Sqrt(Math.Abs(a * x));
        else if ((x > 0 && x <= b) || a != 0)
            f = a * Math.Sin(b * x);
        else
            f = 0;
       Console.WriteLine("x= {0,5:f2}, f(x)= {1,8:f4}",x,f); //вывод
 } }
catch(Exception e)
{            Console.WriteLine("Ошибка: {0}", err.Message);          }
Console.ReadKey();      }
```

Схема алгоритма программы представлена на рисунке 6.2.

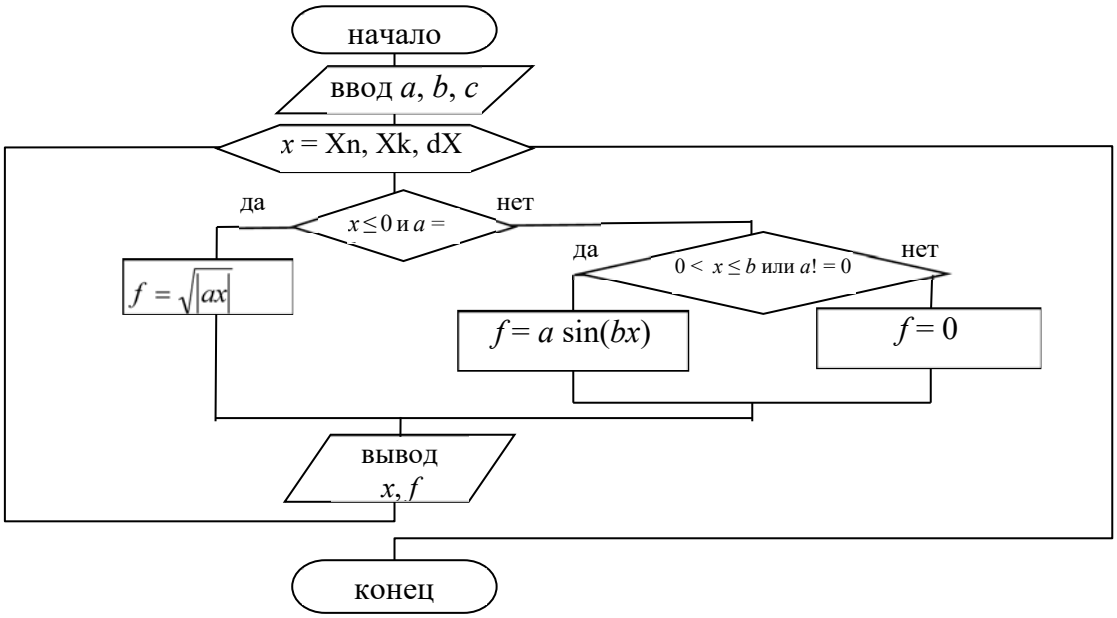


Рисунок 6.2 – Схема алгоритма программы

***Пример 2* (*C*++)** – Вычислить значение конечной суммы: $\frac{1}{1}+\frac{1}{2}+\cdots+\frac{1}{n}$.

В данной задаче:
– исходные данные – целое число n (количество слагаемых);
– результат – вещественное число S (сумма).

```
#include <iostream>
using namespace std;
int main()
{system("chcp 1251");
 int i,n;
 float S = 0;
 cout << "Введите количество слагаемых: ";
 cin >> n;
 for (i = 1; i <= n; i++)
       S += 1.0 / i;
 cout << "S = " << S << endl;
 return 0; }
```

Задание 1

Циклический вычислительный процесс: табулирование функции. Построить таблицу значений для функции $f(x)$ на заданном отрезке [a, b] с числом разбиений отрезка m. Функция $f(x)$, значения a, b, m выбирается из таблицы 6.1.

Таблица 6.1 – Варианты к заданию 1

Номер варианта	$f(x)$	[a, b]	m	Номер варианта	$f(x)$	[a, b]	m
1	$f(x)=\cos(x)$	[0,π/2]	10	7	$f(x)=\arccos(x)$	[0.5,1]	10
2	$f(x)=\sin(x)$	[π/4, π/2]	15	8	$f(x)=\operatorname{arctg}(x)$	[2,7]	15
3	$f(x)=x\cos(x)$	[π/3,2π/3]	20	9	$f(x)=\sin(x)-\cos(x)$	[0,π/2]	20
4	$f(x)=\operatorname{tg}(x)$	[0, π/4]	10	10	$f(x)=x\sin(x)$	[0,3π]	10
5	$f(x)=\operatorname{ctg}(x)$	[π/4,π/2]	15	11	$f(x)=\sin(1/x)$	[π/8,2/π]	15
6	$f(x)=\arcsin(x)$	[0,1]	20	12	$f(x)=x-\sin(x)$	[1,3π/2]	20

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 конечного произведения: $(1+1)\left(1+x^2\right)\cdots\left(1+x^{2n}\right)$.

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: $\frac{7}{9}\cdot\frac{26}{28}\cdots\frac{n^3-1}{n^3+1}$.

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

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

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

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

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

7 Лабораторная работа № 7. Программирование циклических алгоритмов. Операторы while, do ... while

Цель работы:

– изучение циклических операторов языков C#/C++;

– 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.

7.1 Общие сведения

Цикл while. Оператор while проверяет условие завершения цикла перед выполнением тела цикла:

```
i = 0;
while (i < 10)
{   Console.Write("{0} ", i);
    i++;            }
```

В отличие от оператора **for** оператор **while** никак не изменяет значения переменной цикла, поэтому мы должны позаботиться об этом сами.

Цикл do. Оператор **do** используется вместе с ключевым словом **while**. При этом условие завершения цикла проверяется после выполнения его тела:

```
i = 0;
do
{   Console.Write("{0}", i);
    i++;
} while (i < 10);
```

Как только это значение достигнет 10, цикл будет прерван.

Прерывание цикла. С помощью оператора **break** можно в любой момент прервать выполнение цикла. Например, в следующем фрагменте программы прерывается работа цикла, когда значение переменной *i* становится больше пяти:

```
for (i = 0; i < 10; i++)
{   if (i > 5) break;
    Console.Write(" {0} ", i);    }
```

В результате на консоль будут выведены цифры от 0 до 5: 0 1 2 3 4 5.

Возобновление цикла. Оператор **continue** в отличии от **break**, не прекращает выполнение цикла, а переходит к следующей итерации цикла, пропуская оставшийся код цикла. Например, для вывода только положительных чисел:

```
do { x=int.Parse(Console.ReadLine());
     if(x<0) continue;
     Console.Write(" {0} ", x);
} while(x!=100);
```

***Пример 1* (*C#*)** – 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 (схема алгоритма представлена на рисунке 7.1):

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.

Алгоритм решения задачи представлен на рисунке 7.1.

Текст программы (C#):

```
static void Main(string[] args)
 {
  int n, sum;
  Console.WriteLine("Введите число:");
  n=int.Parse(Console.ReadLine());
  sum = 0;
  while (n > 0) //условие цикла
  {
   //увеличиваем sum на последюю
   //цифру
   sum += n % 10;
   //отбрасываем последнюю цифру
   n /= 10;
  }
  Console.WriteLine("Сумма цифр : ", sum);
  Console.ReadKey();
 }
```

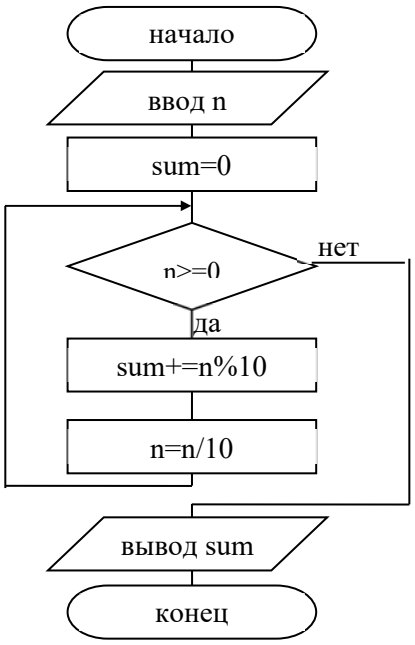


Рисунок 7.1 – Схема алгоритма программы

Пример 2 **(*C*++)** – Написать программу вычисления значения функции cos(x) с помощью бесконечного ряда Тейлора с точностью ε по формуле

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

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. Найдем его:

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

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

Алгоритм решения задачи представлен на рисунке 7.2.

Текст программы:

```
#include <iostream>
using namespace std;
int main()
{ system("chcp 1251");
   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<<"cos x=: "<<y<<endl;
     cout<<"Для
проверки:"<<cos(x)<<endl;
   }
   return 0;
}
```

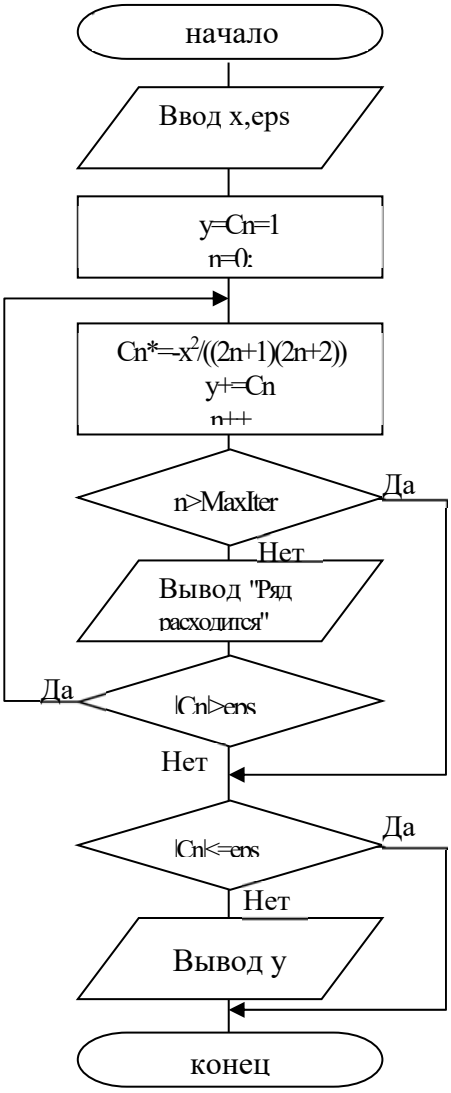


Рисунок 7.2 – Схема алгоритма программы

Задание

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 $\varepsilon = 0{,}0001$. $1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^n} + \cdots$.

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

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

6 Вычислить приближенное значение бесконечной суммы с точностью до ε = 0,0001. $\frac{1}{3\cdot 5}+\frac{1}{7\cdot 9}+\cdots+\frac{1}{(4n-1)(4n+1)}+\cdots$.

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. $1-\frac{1}{2^4}+\frac{1}{3^4}-\cdots\pm\frac{1}{n^4}\mp\cdots$.

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. $\frac{1}{1\cdot 4}+\frac{1}{4\cdot 7}+\cdots+\frac{1}{(3n-2)(3n+1)}+\cdots$.

8 Лабораторная работа № 8. Программирование циклических алгоритмов. Вложенные циклы

Цель работы:

- повторение циклических операторов языков C#/C++;
- изучение операторов перехода.

Постановка задачи. Разработать программы вложенных циклических процессов: вывода последовательности чисел в заданном виде; - вычисление значений функции путем разложения в ряд.

8.1 Общие сведения

***Пример* (*C#*)** – Вывести на экран числа следующим образом:

```
    1   1   1   1   1   1
    2   2   2   2   2   2
    3   3   3   3   3   3
    4   4   4   4   4   4
using System;
namespace Hello
{    class Program
    {      static void Main()
            {
             for (int i = 1; i<=4; ++i, Console.WriteLine())
                 for(int j=1; j<=6; ++j)  Console.Write(" " + i);
} } }
```

Задание 1

Вывести на экран числа следующим образом:

1)
```
41  42  43  …  50
51  52  53  …  60
61  62  63  …  70
…
71  72  73  …  80
```

2)
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

3)
```
8 8 8 8 8
7 7 7 7
6 6 6
5 5
4
```

4)
```
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
```

5)
```
5
5 5
5 5 5
5 5 5 5
5 5 5 5 5
```

6)
```
6 6 6 6 6
7 7 7 7
8 8 8
9 9
10
```

7)
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

8)
```
4 3 2 1 0
3 2 1 0
2 1 0
1 0
0
```

9)
```
1 1 1 1 1
1 1 1 1
1 1 1
1 1
1
```

10)
```
7
6 6
5 5 5
4 4 4 4
3 3 3 3 3
```

11)
```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

12)
```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

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. 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) $\arcsin x = x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \ldots \cdot 2n \cdot (2n+1)} = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} - \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \cdots \quad |x| < 1.$

9 Лабораторная работа № 9. Обработка одномерных массивов. Сортировка массивов

Цель работы:

– получение практических навыков в работе с одномерными массивами;

– знакомство 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.

9.1 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. Нумерация элементов массива начинается с нуля, т. е., если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерные массивы. Объявления одномерного массива в C# может использоваться одна из следующих форм записи (таблица 9.1).

Таблица 9.1 – Объявления одномерного массива в C#

Форма записи	Пояснения
базовый_тип [] имя_массива = new базовый_тип [размер]; *Например*: `int []a=new int [10];`	Объявлен одномерный массив заданного типа и выделена память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. В C# элементам массива присваиваются начальные значения по умолчанию: для арифметических типов – нули, для ссылочных типов – null, для символов – пробел
базовый_тип [] имя_массива= {список инициализации}; *Например:* `int []a={0, 1, 2, 3};`	Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации

Обращения к элементам массива происходит с помощью индекса: имя массива и в квадратных скобках его номер. Например, *a*[0], *b*[10], *c*[*i*].

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Рассмотрим несколько простых примеров работы с одномерными массивами.

```
int[] myArray = {0,1,2,3,4,5,6,7,8,9};
int i;
for (i = 0; i < 10; ++i)
    Console.WriteLine(myArray[i]);
```

```
int[] myArray = new int[10];
int i;
for (i = 0; i < 10; i++) myArray[i] = i*i;
for (i = 0; i < 10; i++)
   Console.WriteLine(myArray[i]);
```

Стандартный синтаксис объявления одномерного массива в C++:

```
тип имя_массива[размер];
```

где *тип* объявляет базовый тип массива и является типом каждого элемента массива, *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:

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

В языке C++ отсутствует проверка границ массивов.

Пример (C++) – 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, т. е. его индекс:

1) задать начальные значения для индексов минимального элемента (например, равные нулю);

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 (т. е. запомнить его индекс).

Для обмена необходимо использовать дополнительную переменную. Процесс обмена проиллюстрируем на рисунке 9.1.

Алгоритм решения задачи представлен на рисунке 9.2.

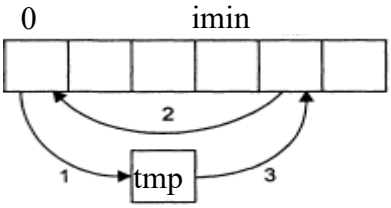


Рисунок 9.1 – Механизм обмена элементов массива

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

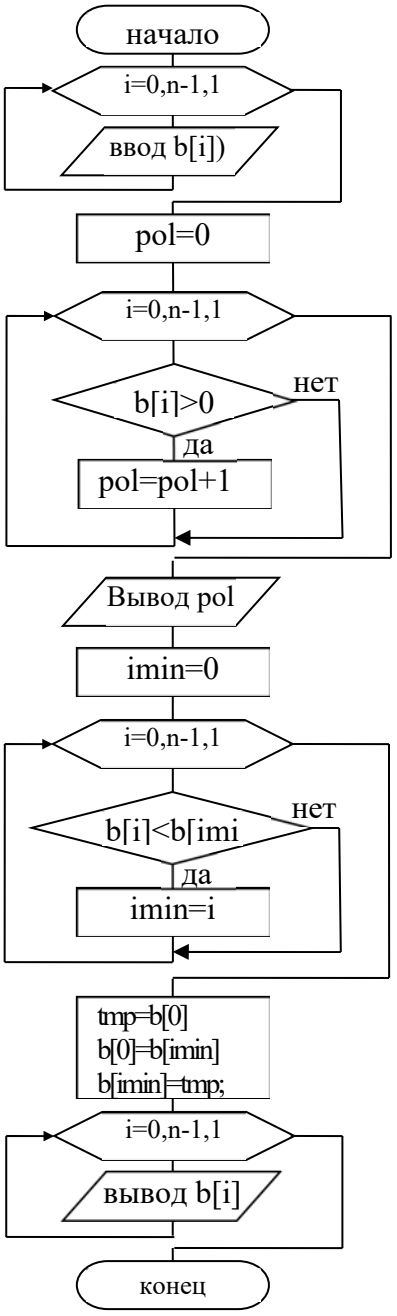


Рисунок 9.2 – Схема алгоритма программы

Сортировка. Рассмотрим массив целых или вещественных чисел $a_1, a_2, \ldots, a_n$. 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 т. д. (Сортировка выбором.)

2 Последовательным просмотром чисел $a_1, a_2, \ldots, a_n$ 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 т. д. 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 $a_2, \ldots, a_n$ и каждый новый элемент вставлять на подходящее место в уже упорядоченную последовательность $a_1, \ldots, a_{i-1}$. Это место определяется последовательным сравнением a_i с упорядоченными элементами $a_1, \ldots, a_{i-1}$. (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. Далее сравнить элементы a_2 и a_3 и, если $a_2 > a_3$ (или $a_2 < a_3$), то их переставить. Далее сравнить элементы a_3 и a_4 и так далее до элементов a_{n-1} и a_n включительно. Далее эти действия повторить, начиная опять с первого элемента. Последним является контрольный проход, при котором не будет перестановок элементов. (Сортировка по методу пузырька.)

Задания

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; в) 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; в) 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; в) 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; в) 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; в) 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; в) 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; в) 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; в) 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; в) 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; в) 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 элементов, вычислить: а) номер минимального по модулю элемента массива; б) сумму модулей элементов массива, расположенных после первого отрицательного элемента; в) 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; в) преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале $[a, b]$, а потом – все остальные.

10 Лабораторная работа № 10. Обработка массивов как объектов

Цель работы:

– получение практических навыков в работе с одномерными массивами;

– знакомство с классом Array в C#, динамическими массивами в C++.

Постановка задачи. Задать массив исходных данных случайными числами и выполнить над ним указанные действия с использованием класса Array (на C++ с помощью динамических массивов).

10.1 Общие сведения

Массив как объект. Массивы в C# могут быть реализованы как объекты. Если говорить более точно, то они реализованы на основе базового класса Array, определенного в пространстве имен System. Данный класс содержит различные свойства и методы, основные приведены в таблице 10.1.

Таблица 10.1 – Объявления одномерного массива в C#

Элемент	Вид	Описание
Length	Свойство	Количество элементов массива (по всем размерностям)
BinarySearch	Статический	Двоичный поиск в отсортированном массиве
Clear	Статический	Присваивание элементам массива значений по умолчанию
Copy	Статический	Копирование заданного диапазона элементов одного массива в другой
CopyTo	Экземплярный	Копирование всех элементов текущего одномерного массива в другой
GetValue	Экземплярный	Получение значения элемента массива
IndexOf	Статический	Поиск первого вхождения элемента в одномерный массив
LastIndexOf	Статический	Поиск последнего вхождения элемента в одномерный массив
Reverse	Статический	Изменение порядка следования элементов на обратный
SetValue	Экземплярный	Установка значения элемента массива
Sort	Статический	Упорядочивание элементов одномерного массива

Вызов *статических* методов происходит через обращение к имени класса, например, Array.Sort(myArray). Обращение к свойству или вызов *экземплярного* метода производится через обращение к экземпляру класса, например, myArray.Length или myArray.GetValue(i).

Пример

```
static void Main()
{try
  {   Console.Write("Введите размерность массива: ");
      int n = int.Parse(Console.ReadLine());
      int[] MyArray = new int[n];
      for (int i = 0; i < MyArray.Length; ++i)
         {   Console.Write("a[{0}]=", i);
             MyArray[i] = int.Parse(Console.ReadLine());      }
      PrintArray("исходный массив:", MyArray);
      Array.Sort(MyArray);
      PrintArray("массив отсортирован по возрастанию", MyArray);
      Array.Reverse(MyArray);
      PrintArray("массив отсортирован по убыванию", MyArray);
  }
```

```
 catch (Exception)
  { Console.WriteLine("Ошибка"); }
 Console.ReadKey();
}
        static void PrintArray(string a, int[] mas)
        {   Console.WriteLine(a);
            for (int i = 0; i < mas.Length; i++) Console.Write("{0} ", mas[i]);
            Console.WriteLine();

        }
```

В языке C++ мы можем использовать динамические массивы, размер которого можно задавать во время выполнения программы. Для выделения памяти под динамический массив также используется оператор new, после которого в квадратных скобках указывается, сколько массив будет содержать объектов:

```
int *numbers = new int[4]; // динамический массив из 4 чисел
// или так    int *numbers {new int[4]};
```

Причем в этом случае оператор new также возвращает указатель на объект типа int – первый элемент в созданном массиве. В данном случае определяется массив из четырех элементов типа int, но каждый из них имеет неопределенное значение. Однако также можно инициализировать массив значениями:

```
int *numbers1 = new int[4]{};         // массив состоит из чисел 0, 0, 0, 0
int *numbers2 = new int[4]{ 1, 2, 3, 4 };// массив состоит из чисел 1, 2, 3, 4
int *numbers3 = new int[4]{ 1, 2 };       // массив состоит из чисел 1, 2, 0, 0
```

При инициализации массива конкретными значениями следует учитывать, что если значений в фигурных скобках больше, чем длина массива, то оператор new потерпит неудачу и не сможет создать массив.

После создания динамического массива можно с ним работать по полученному указателю, получать и изменять его элементы:

```
int *numbers {new int[4]{ 1, 2, 3, 4 }};
// получение элементов через синтаксис массивов
cout << numbers[0] << endl << numbers[1] << endl;
// получение элементов через операцию разыменования
cout << *numbers << endl << *(numbers+1) << endl;
```

Причем для доступа к элементам динамического массива можно использовать как синтаксис массивов (numbers[0]), так и операцию разыменования (*numbers).

Задание

Варианты задания взять из лабораторной работы № 9. Вариант задания для лабораторной работы № 10 выбирается по правилу 13–*N*, где *N* – номер варианта.

11 Лабораторная работа № 11. Двумерные массивы

Цель работы:

– изучение двумерных и ступенчатых массивов;

– закрепление навыков структурного программирования.

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

11.1 Основные сведения

Многомерные массивы. В языке программирования C# многомерные массивы объявляются с использованием запятых для разделения размерностей. Общий синтаксис объявления многомерного массива выглядит следующим образом:

```
<тип>[, ... ,] < имя_массива>;
```

Число запятых указывает на размерность массива. Объявители (имя_массива) следуют тем же правилам, что и для одномерных массивов.

Хотя можно явно инициализировать многомерные массивы с использованием константных значений, это редко используется из-за громоздкости такой структуры. Обычно инициализацию многомерных массивов производят программно. Например,

```
int[,] matrix = { { 1, 2 }, { 3, 4 } }; // матрица из 2 строк и 2 столбцов
int [,] dM;
Console.WriteLine("Введите размеры матрицы:");
int n = Convert.ToInt32(Console.ReadLine());
int m = Convert.ToInt32(Console.ReadLine());
dM = new int[n,m];
Console.WriteLine("Введите элементы матрицы:");
for(int i = 0;i <n; i++)
    for(int j = 0;j <m; j++)   dM[i, j] = Convert.ToInt32(Console.ReadLine());
int max = dM[0, 0];
for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
       if (dM[i, j] > max) max = dM[i, j];
Console.WriteLine("Max. element - {0}", max);
for (int i = 0; i < n; i++)
{  for (int j = 0; j < m; j++) Console.Write("{0} ", dM[i, j]);
   Console.WriteLine();    }
```

В языке C++ двумерный массив объявляется следующим образом:

```
<тип> <имя_массива>[<размер1>][<размер2>];
```

Следовательно, для объявления двумерного массива целых чисел с размером 5 и 8 следует написать: `int  d[5][8];`

В противоположность C#, 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:

```
#include <iostream>
using namespace std;
int main()
{  system("chcp 1251");
   int  i, j;
   int num[3][4];
   for (i=0; i<3; i++)
   {   cout<< "Введите элементы"<<i+1<< "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.

Массивы массивов. Еще одним видом *массивов* C# являются ***массивы массивов***, называемые также ***изрезанными* (*ступенчатыми*) *массивами* (*jagged arrays*)**. Такой *массив массивов* можно рассматривать как одномерный *массив*, элементы которого являются *массивами*, и так может продолжаться до некоторого уровня вложенности, например

```
int[][] jagger = new int[3][]
{  new int[] {5,7,9,11},
   new int[] {2,8},
   new int[] {6,12,4}    };
```

Массив jagger имеет всего два уровня. Можно считать, что у него три элемента, каждый из которых является *массивом*. Для каждого такого *массива* необходимо вызвать конструктор new, чтобы создать внутренний *массив*. В данном примере элементы внутренних *массивов* получают значение, будучи явно инициализированы константными *массивами*. Конечно, допустимо и такое объявление:

```
int[][] jagger1 = new int[3][] { new int[4], new int[2], new int[3] };
```

В этом случае элементы *массива* получат при инициализации нулевые значения. Реальную инициализацию нужно будет выполнять программным путем. Стоит заметить, что в конструкторе верхнего уровня константу 3 можно опустить и писать просто new int[][].

Аналогом в C++ являются многомерные динамические массивы. Рассмотрим на примере двухмерных массивов. По сути двухмерный массив – это набор массив массивов. Соответственно, чтобы создать динамический двухмерный массив, нам надо создать общий динамический массив указателей, а затем его элементы – вложенные динамические массивы. В общем случае это выглядит так:

```
unsigned rows = 3;       // количество строк
unsigned columns = 2;    // количество столбцов
```

```
int** numbers{new int*[rows]{}};  // выделяем память под двухмерный массив
// выделяем память для вложенных массивов
for (unsigned i{}; i < rows; i++) numbers[i] = new int[columns]{};
// удаление массивов
for (unsigned i{}; i < rows; i++) delete[] numbers[i];
delete[] numbers;
```

Задание 1

1 Дана целочисленная прямоугольная матрица. Определить количество строк, не содержащих ни одного нулевого элемента.

2 Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих ни одного нулевого элемента.

3 Дана целочисленная квадратная матрица. Определить произведение элементов в тех строках, которые не содержат отрицательных элементов.

4 Дана целочисленная квадратная матрица. Определить сумму элементов в тех столбцах, которые не содержат отрицательных элементов.

5 Дана матрица размера 5 × 4. Написать программу для вычисления *I* нормы матрицы: $|A|_I = \max_{i=1,n} \sum_{k=1}^{n} |a_{ik}|$.

6 Найти наибольший элемент матрицы *A*(5 × 3) и номер строки и столбца, в котором он находится.

7 Вычислить сумму элементов каждой строки матрицы *X* (4 × 3), определить наименьшее значение этих сумм и номер соответствующей строки.

8 Найти наибольшие элементы каждой строки матрицы *X* (4 × 5*)* и записать их в массив Y.

9 Найти наибольший элемент главной диагонали матрицы *A* (4 × 4*)* и вывести на экран все строку, в которой он находится.

10 В массиве *M* [6 × 4] все числа различны. В каждой строке находится минимальный элемент, затем среди этих чисел выбирается максимальное. Напечатать номер строки массива *M*, в которой расположено выбранное число.

Задание 2

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

1 Дана целочисленная прямоугольная матрица. Определить: количество строк, не содержащих ни одного нулевого элемента; 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: количество столбцов, содержащих хотя бы один нулевой элемент; номер строки, в которой находится самая длинная серия одинаковых элементов.

4 Дана целочисленная квадратная матрица. Определить: произведение элементов в тех строках, которые не содержат отрицательных элементов; максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

5 Дана целочисленная квадратная матрица. Определить: сумму элементов в тех столбцах, которые не содержат отрицательных элементов; минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

6 Дана целочисленная прямоугольная матрица. Определить: сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент; номера строк и столбцов всех седловых точек матрицы. 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 Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.

12 Лабораторная работа № 12. Символьные и строковые типы

Цель работы:

- изучение символьных и строковых типов;
- использование строковых типов.

Постановка задачи. Для конкретного варианта реализовать задачи с использованием типов char, string и класса StringBuilder.

12.1 Основные сведения

Класс char. В C# есть *символьный тип* ***char***, основанный на классе System.Char и использующий двухбайтную кодировку Unicode представления символов. Методы и свойства класса Char представлены в таблице 12.1.

Класс string. Основным типом при работе со строками является тип *string*, задающий строки переменной длины.

Таблица 12.1 – Статические методы и свойства класса Char

Метод	Описание
GetNumericValue	Возвращает численное значение символа, если он является цифрой, и (-1) в противном случае
GetUnicodeCategory	Метод возвращает Unicode категорию символа.
IsControl	Возвращает true, если символ является управляющим
IsDigit	Возвращает true, если символ является десятичной цифрой
IsLetter	Возвращает true, если символ является буквой
IsLetterOrDigit	Возвращает true, если символ является буквой или цифрой
IsLower	Возвращает true, если символ задан в нижнем регистре
IsNumber	Возвращает true, если символ является числом (десятичной или шестнадцатиричной цифрой)
IsPunctuation	Возвращает true, если символ является знаком препинания
IsSeparator	Возвращает true, если символ является разделителем
IsUpper	Возвращает true, если символ задан в верхнем регистре
IsWhiteSpace	Возвращает true, если символ является «белым пробелом». К белым пробелам, помимо пробела, относятся и другие символы, например, символ конца строки и символ перевода каретки
Parse	Преобразует строку в символ. Строка должна состоять из одного символа, иначе возникнет ошибка
ToLower	Приводит символ к нижнему регистру
ToUpper	Приводит символ к верхнему регистру
MaxValue, MinValue	Свойства, возвращающие символы с максимальным и минимальным кодом. Возвращаемые символы не имеют видимого образа

Объекты *класса string* объявляются как все прочие объекты простых типов – с явной или отложенной инициализацией. Например, объявления строк с вызовом разных конструкторов:

```
string world = "Мир";
string sss = new string('s', 5);
char[] yes = "Yes".ToCharArray();
string syes = new string(yes);
Console.WriteLine("world = {0}; sss={1}; syes={2};", world, sss, syes);
```

Объект world создан без явного вызова конструктора, а объекты sss, syes, sye созданы разными конструкторами *класса String*.

Операции над строками: присваивание (=); две операции проверки эквивалентности (= =) и (!=); конкатенация или сцепление строк (+); взятие индекса ([]).

Вот пример, в котором над строками выполняются данные операции:

```
string s1 = "ABC", s2 = "CDE";
string s3 = s1 + s2;
bool b1 = (s1 == s2);
char ch1 = s1[0], ch2 = s2[0];
Console.WriteLine("s1={0}, s2={1}, b1={2}, ch1={3}, ch2={4}", s1,s2,b1,ch1,ch2);
```

В таблице 12.2 приведены основные методы и свойства работы со троками.

Методы Join и Split. Методы *Join* и *Split* выполняют над строкой текста взаимно обратные преобразования. *Динамический метод Split* позволяет осуществить разбор текста на элементы. *Статический метод Join* выполняет обратную операцию, собирая строку из элементов.

Таблица 12.2 – Статические методы и свойства класса String

Метод	Описание
Empty	Возвращается пустая строка. Свойство со статусом read only
Compare	Сравнение двух строк. Реализации метода позволяют сравнивать как строки, так и подстроки. При этом можно учитывать или не учитывать регистр, особенности национального форматирования дат, чисел и т. д.
CompareOrdinal	Сравнение двух строк. Метод перегружен. Реализации метода позволяют сравнивать как строки, так и подстроки. Сравниваются коды символов
Concat	Конкатенация строк, допускает сцепление произвольного числа строк
Copy	Создается копия строки
Format	Форматирование в соответствии с заданными спецификациями формата

Рассмотрим примеры применения этих методов. В первом из них строка разбивается на простые предложения. Во втором предложение разделяется на слова. Затем производится обратная сборка разобранного текста. Код соответствующей процедуры:

```
string txt = "А это пшеница, которая в темном чулане хранится, в доме, который построил Джек!";
Console.WriteLine("txt={0}", txt);
Console.WriteLine("Разделение текста на простые предложения:");
  string[] SimpleSentences, Words;
  //размерность массивов SimpleSentences и Words устанавливается автоматически в
  // соответствии с размерностью массива, возвращаемого методом Split
  SimpleSentences = txt.Split(',');
  for (int i = 0; i < SimpleSentences.Length; i++)
   Console.WriteLine("SimpleSentences[{0}]= {1}", i, SimpleSentences[i]);
  string txtjoin = string.Join(",", SimpleSentences);
  Console.WriteLine("txtjoin={0}", txtjoin);
  Words = txt.Split(',', ' ');
  for (int i = 0; i < Words.Length; i++)
   Console.WriteLine("Words[{0}]= {1}", i, Words[i]);
  txtjoin = string.Join(" ", Words);
  Console.WriteLine("txtjoin={0}", txtjoin);
```

Динамические методы класса String. Рассмотрим наиболее характерные методы при работе со строками (таблица 12.3). Следует помнить, что *класс string* является *неизменяемым*. Поэтому Replace, Insert и другие методы представляют собой функции, возвращающие новую строку в качестве результата и не изменяющие строку, вызвавшую метод.

В C++ строка представляет собой массив символов, заканчивающийся нуль-символом. Нуль-символ – это символ с кодом, равным 0, т. е. '\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. Например,

```
const int n=80;
char str[n];  //объявление  строки
```

```
cin.getline(str,n);  //ввод строки
cout<<s<<endl;  //вывод строки
```

Таблица 12.3 – Динамические методы и свойства класса String

Метод	Описание
Insert	Вставляет подстроку в заданную позицию
Remove	Удаляет подстроку в заданной позиции
Replace	Заменяет подстроку в заданной позиции на новую подстроку
Substring	Выделяет подстроку в заданной позиции
IndexOf, IndexOfAny, LastIndexOf, LastIndexOfAny	Определяются индексы первого и последнего вхождения заданной подстроки или любого символа из заданного набора
StartsWith, EndsWith	Возвращается true или false, в зависимости от того, начинается или заканчивается строка заданной подстрокой
PadLeft, PadRight	Добавление нужного числа пробелов в начале и в конце строки
Trim, TrimStart, TrimEnd	Обратные операции к методам Pad. Удаляются пробелы в начале и в конце строки, или только с одного ее конца
ToCharArray	Преобразование строки в массив символов

Функции для работы со строками в C++ находятся в библиотеке string.h и приведены в таблице 12.4.

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 *Str*1 на подстроку *Str*2 (подстроки вводятся с клавиатуры)

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

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

Таблица 12.4 – Функции работы со строками в C++

Действие	Вид функции	Пример использования

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;`

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");`

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	`char s1[80],s2[80];`
		`cin.getline(s1,80);`
		`cin.getline(s2,80);`
		`strcat(s2,s1);`
		`cout<<s2<<endl;`

Сравнение строк	**char *strcmp (char *str1, char *str2)** выполняет алфавитное сравнение двух строк и возвращает целое число:
	Число **Значение**
	Меньше 0 *str*1 меньше, чем *str*2
	Равно 0 *str*1 равна *str*2
	Больше 0 *str*1 больше, чем *str*2
		Следующий код используется для проверки пароля:
		`char s[80];`
		`cout<<"Введи пароль: ";`
		`cin.getline(s,80);`
		`if(strcmp(s, "pass")) {`
		`cout<<"Неправильный пароль. ";`

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**	`char *p;`
		`p=strchr("this is a test",' ');`
		`cout<<p<<endl;`

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";`

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;`
		`char 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 2

1 Дан символ *C*. Вывести его код (то есть номер в кодовой таблице).

2 Дано целое число N ($32 \le N \le 126$). Вывести символ с кодом, равным N.

3 Дан символ C. Вывести два символа, первый из которых предшествует символу C в кодовой таблице, а второй следует за символом *C*.

4 Дано целое число N ($1 \le N \le 26$). Вывести N первых прописных (то есть заглавных) букв латинского алфавита.

5 Дано целое число N ($1 \le N \le 26$). Вывести N последних строчных (то есть маленьких) букв латинского алфавита в обратном порядке (начиная с буквы «z»).

6 Дан символ *C*, изображающий цифру или букву (латинскую или

русскую). Если *C* изображает цифру, то вывести строку «digit», если латинскую букву – вывести строку «lat», если русскую — вывести строку «rus».

7 Дана непустая строка. Вывести коды ее первого и последнего символа.

8 Дано целое число *N* и символ *C*. Вывести строку длины *N*, которая состоит из символов *C*.

9 Дано четное число *N* и символы *C*1 и *C*2. Вывести строку длины *N*, которая состоит из чередующихся символов *C*1 и *C*2, начиная с *C*1.

10 Дана строка. Вывести строку, содержащую те же символы, но расположенные в обратном порядке.

11 Дана непустая строка *S*. Вывести строку, содержащую символы строки S, между которыми вставлено по одному пробелу.

12 Дана непустая строка *S* и целое число *N*. Вывести строку, содержащую символы строки *S*, между которыми вставлено по *N* символов «*».

Задание 3

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

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

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

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

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

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

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

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

8 Подсчитать сколько раз заданное слово встречается в строке.

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

10 Найти самое длинное слово строки.

11 Найти все самые длинные слова строки.

12 Найти самое короткое слово строки.

Список литературы

1 **Гуриков, С. Р.** Введение в программирование на языке Visual C#: учебное пособие / С. Р. Гуриков. – Москва: ФОРУМ; ИНФРА-М, 2020. – 447 с.

2 **Дадян, Э. Г.** Современные технологии программирования. Язык C#: учебник: в 2 т. / Э. Г. Дадян. – Москва : ИНФРА-М, 2021. – Т. 1. – 312 с.

3 **Гагарина, Л. Г.** Технология разработки программного обеспечения: учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул; под ред. Л. Г. Гагариной. – Москва: ФОРУМ; ИНФРА-М, 2019. – 400 с.

4 **Немцова, Т. И.** Программирование на языке высокого уровня. Программирование на языке C++: учебное пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев; под ред. Л. Г. Гагариной. – Москва: ФОРУМ; ИНФРА-М, 2021. – 512 с.