МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» дневной и заочной форм обучения

Часть 1

Могилев 2019

УДК 004.4
БКК 32.973
О 75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «1» сентября 2017 г., протокол № 1

Составитель ст. преподаватель А. И. Кашпар

Рецензент _____

Методические рекомендации к лабораторным работам предназначены для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» дневной и заочной форм обучения.

Учебно-методическое издание

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

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

Подписано в печать . Формат 60x84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл.- печ. л. . Уч.-изд. л. . Тираж 31 экз. Заказ №

Издатель и полиграфическое исполнение:
«Белорусско-Российский университет»
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилёв,

Содержание

Введение

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

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

Методические указания содержат цикл лабораторных работ в объёме 34 часов аудиторных занятий, предназначенный для выполнения в осеннем семестре студентами первого курса дневной формы обучения специальности 1-53 01 02 «Автоматизированные системы обработки информации». Для студентов заочной формы обучения в осеннем семестре обязательны к выполнению 2, 4, 6, 7, 9 и 11.

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

Лабораторная работа№ 1. Тема «Алгоритмы»

Цель работы

- программирование базовых конструкций алгоритмов;
- получение практических навыков по работе с алгоритмами.

Постановка задачи

Составить блок схемы для четырех заданий.

Общие сведения

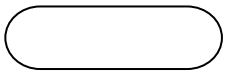
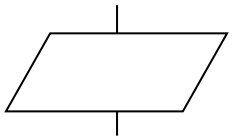
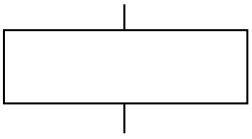
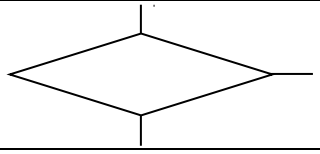
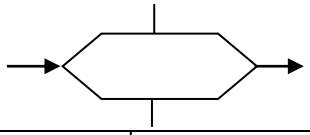
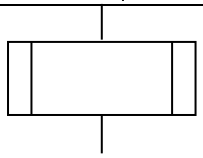
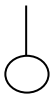
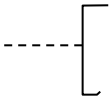
1 Алгоритмизация

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

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

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

Таблица 1.2 – Знаки и символы, используемые в алгоритмах

«да», «нет»; «+», «–»; «1», «0».	Условие истинно: «да», «+», «1». Условие ложно: «нет», «–», «0.

Стандартное ветвление имеет два вероятных направления

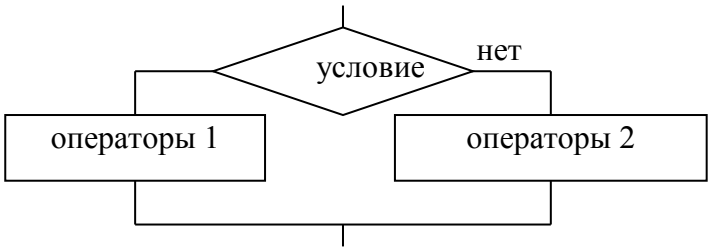


Рисунок 1.1 – Стандартное ветвление

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

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

Тело цикла – группа повторяющихся инструкций.

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

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

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

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

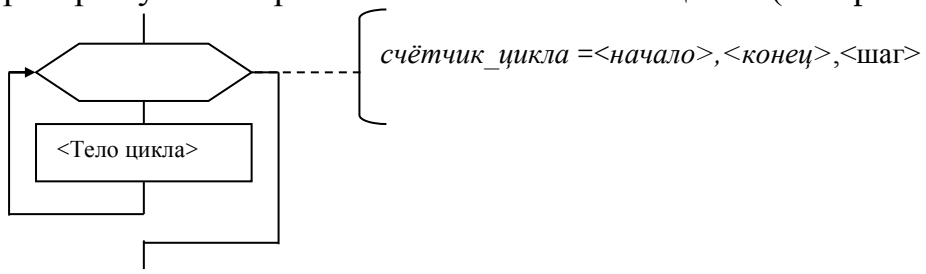


Рисунок 1.2 – Циклический алгоритм с параметром

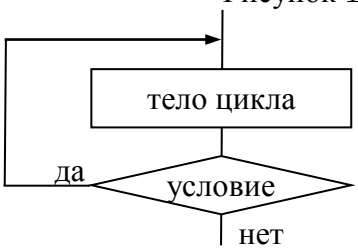


Рисунок 1.3 – Циклический алгоритм с постусловием

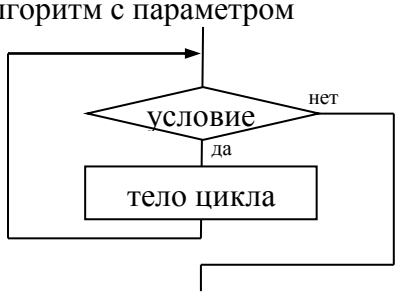


Рисунок 1.4 – Циклический алгоритм предусловием

2 Примеры алгоритма

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.

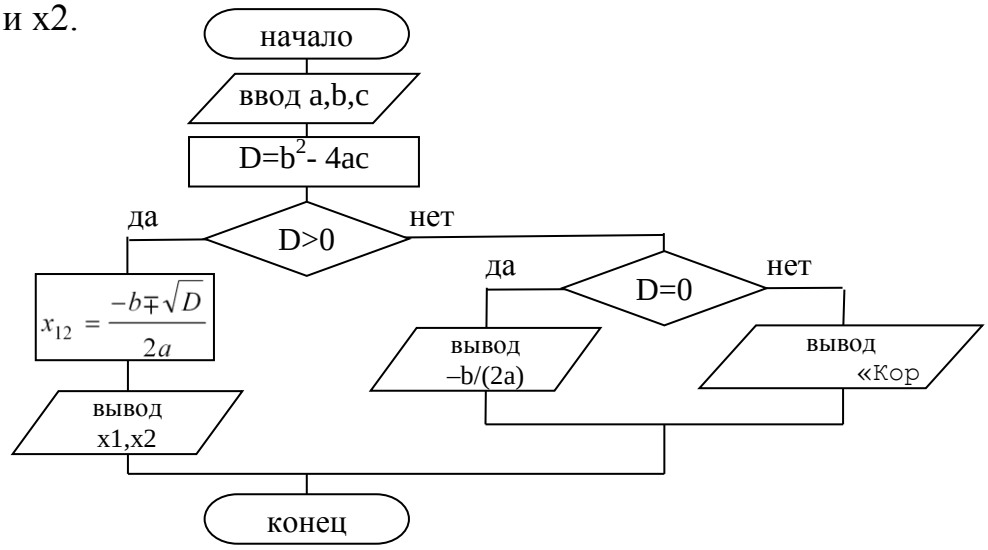


Рисунок 1.5 – Пример составления разветвляющего алгоритма

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

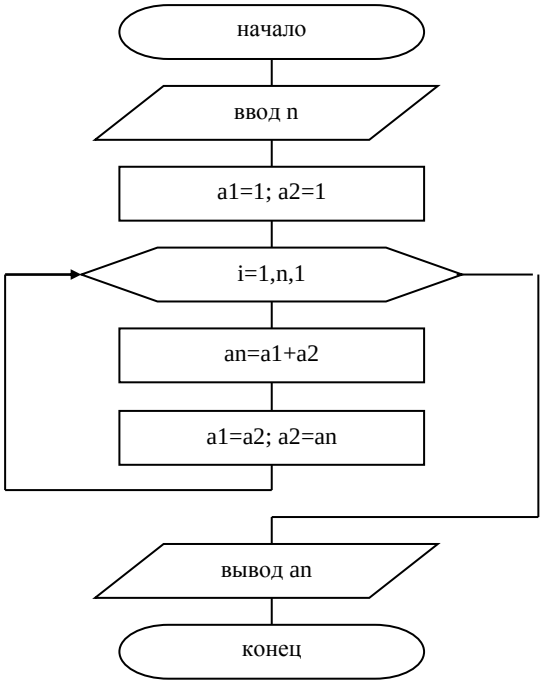


Рисунок 1.6 – Пример составления циклического алгоритма

Задание 1.

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

Исходные данные к заданию находятся в таблице 1.3.

Таблица 1.3 - Индивидуальные задания

№	Данные1	Данные 2	Переменная условия	Условие выполняется	Условие не выполняется
1	A_1= 2.35	A_2=-5.89	B=(A_1^2-A_2) >0	C=3.1B-$A_1A_2^2$	C=$A_1^2A_2$-3.1B
2	X_1=-8.44	X_2= 1.73	Y=(4X_1-X_2) <0	Z=5Y^2-X_2+2X_1	Z=X_1+X_2-5Y^2
3	E_1= 7.54	E_2=-3.62	F=(3E_2-E_1) >0	Y=4F /(E_1-E_2)	Y=4F/(E_2-E_1)
4	B_1=-6.71	B_2= 4.57	E=(B_1-B_2^2) <0	X=2E -B_1+B_2^2	X=2E -B_2+B_1^2
5	C_1= 3.26	C_2=-5.41	X=(C_1 C_2) >0	E=7.5X - C_1/C_2	E=C_1/C_2 -7.5X
6	D_1=-9.08	D_2= 6.35	I=(D_1^2-D_2^2) <0	B=2.1D_1-3.4 I	B=1.2D_2-3.4 I
7	F_1= 5.12	F_2=-2.06	A=(7F_1^2-F_2) >0	D=3.1A-7F_1F_2	D=7F_1F_2-3.1A
8	H_1=-0.97	H_2= 7.94	D=(H_1/H_2^2) <0	F=8D-7.5H_1H_2	F=4.5H_1H_2-6D
9	I_1= 1.63	I_2=-9.18	H=(3I_1^2-I_2^2) >0	A=H/(3.8I_1-I_2)	A=H/(I_2-3.8I_1)
10	J_1=-4.32	J_2= 0.79	C=(J_1J_2-9) <0	I=3C-4.1(J_1J_2)^2	I=4(J_1J_2)^2-3C
11	K_1= 7.45	K_2=-3.82	J=(K_1-5K_2^2) >0	H=2.5J-K_1K_2	H=K_1K_2-2.5J
12	L_1=-2.71	L_2= 5.63	K=(L_1^2/L_2) <0	J=9K-3L_1+L_2	J=3L_1-L_2-9K
13	M_1=5.23	M_2=-1.1	L=(M_1-M_2) >0	K=L-5.4M_1M_2	K=5.4M_1M_2-L
14	N_1=-3.64	N_2= 4.47	M=($N_1^2N_2$) <0	L=M/(N_1-N_2)	L=M/(N_2-N_1)
15	P_1= 8.19	P_2=-2.34	N=(2P_1/P_2^2) >0	M=(3P_1-P_2)/N	M=(3P_2-P_1)/N

Задание 2.

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

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

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

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

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

6. Даны три числа. Найти наименьшее из них.

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

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

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

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

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

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

13. Даны четыре целых числа, одно из которых отлично от трех других, равных между собой. Определить порядковый номер числа, отличного от остальных.

14. 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.

15. 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.

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 табл. 1.4.

Таблица 1.4 - Индивидуальные задания

№	Начальное значение	Конечное значение	Значение шага	Выражение для расчета значений функции
1	Xn = -0.25	Xk = 1.75	Hx = 0.05	$Y = 2.23\ X^2 + 3.12\ X - 1.95$
2	Yn = - 2.20	Yk = 2.30	Hy = 0.10	$Z = -1.2\ 5Y^2 + 0.46\ Y - 0.88$
3	Zn = -5.25	Zk = 4.75	Hz = 0.25	$A = 2.85\ Z^2 - 4.15\ Z + 6.05$
4	An = -10.50	Ak = 30.50	Ha = 0.50	$B = -1.45\ A^2 + 9.15\ A + 12.5$
5	Bn = -4.20	Bk = 3.80	Hb = 0.20	$C = 0.52\ B^2 + 2.52\ B + 7.84$
6	Cn = -3.50	Ck = 11.50	Hc = 0.35	$D = -0.15\ C^2 - 5.33\ C - 9.21$
7	Dn = -3.60	Dk = 12.40	Hd = 0.40	$E = 1.08\ D^2 - 7.22\ D - 2.43$
8	En = -0.15	Ek = 0.45	He = 0.15	$F = 4.02\ E^2 - 8.49\ E + 3.14$
9	Fn = - 9.90	Fk = 2.10	Hf = 0.30	$G = -2.36\ F^2 + 6.07\ F + 8.3$
10	Gn = -6.75	Gk = 11.25	Hg = 0.45	$I = -3.44\ G^2 - 4.72\ G + 5.57$
11	In = -13.20	Ik = 10.08	Hi = 0.60	$J = 0.97\ I^2 - 1.75\ I + 4.32$
12	Jn = -9.60	Jk = 22.40	Hj = 0.80	$K = 1.06\ J^2 + 5.67\ J - 6.98$
13	Kn = -13.0	Kk = 13.0	Hk = 0.65	$L = 0.64\ K^2 + 2.71\ K + 7.07$
14	Ln = -11.25	Lk = 26.25	Hl = 0.75	$M = -5.05\ L^2 - 9.18\ L - 11.91$
15	Mn = -14.0	Mk = 14.0	Hm = 0.70	$H = -3.33\ M^2 + 8.54\ M + 9.61$

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 $N^2 + (N + 1)^2 + (N + 2)^2 + \ldots + (2 \cdot N)^2$

3. Дано целое число N > 0. Найти произведение 1*2*3...

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

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

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

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

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

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

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

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

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

13. Дано вещественное число X (|X|<1) и целое число N > 0. Найти значение выражения $X - X^3/3 + X^5/5 - \ldots + (-1)^N \cdot X^{2N+1}/(2 \cdot N+1)$. Полученное число является приближенным значением функции arctg в точке X.

14. 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, A+3H, ..., B, образующий разбиение отрезка [A,B].

15. 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 отрезка, а также значения функции F(X) = 1 - sin(X) в точках, разбивающих отрезок [A,B]: F(A), F(A+H), F(A+2H), ..., F(B).

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

Цель работы

- Изучение интегрированной среды:
- Освоение простейшей структуры программы на языке C#:
- Получение навыков в организации ввода-вывода на языке C#.
- Изучить базовые типы данных языка C#.

Постановка задачи

Напишите программу для расчета по двум формулам.

Общие сведения

1 Линейная программа

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

Задача 1. 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, а 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:

- в качестве исходных данных выступает одно вещественное число, представляющее собой температуру по Цельсию,

- в качестве результата — другое вещественное число - *температура по Фаренгейту*.

Перед написанием программы откроем интегрированную среду Visual Studio:

Пуск/Программы/Microsoft Visual Studio …/ Microsoft Visual C#...

Далее создадим проект. Для этого:

1) File(Файл) > ***"New Project..."*** (Создать проект).

2) В открывшемся окне ***New Project(Создать проект)***:

- выберем "Other languages" -> "Visual C#" -> "Windows"

- выберите тип ***Console Application (Консольное приложение);***

- введите имя проекта в текстовом поле ***Name(Имя)*** *(например Lab2)*;

-в строке ***Location (Расположение)*** определите положение на диске, куда нужно сохранять ваш проект (например Z:\Gruppa\FIO) .

- щелкните левой кнопкой мыши на кнопке ОК.

У нас появиться шаблон консольного приложения, формы, естественно нет, так как это у нас будет консольного приложение. В шаблоне присутствует только заготовка текста программы:

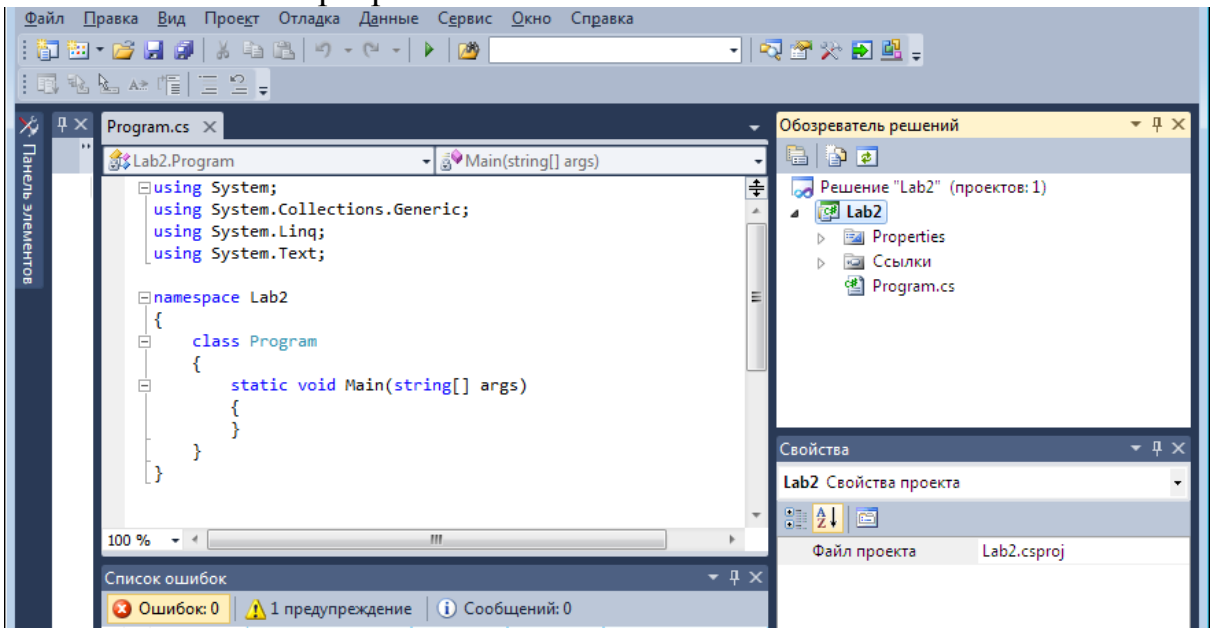


Рисунок 2.1 – Шаблон консольного приложения

В правой верхней части располагается окно управления проектом ***Solution Explorer***. Если оно закрыто, то его можно включить командой ***View - Solution Explorer***. В этом окне перечислены все ресурсы, входящие в проект, например Program.cs - текст программы на языке C#.

В правой нижней части экрана располагается окно свойств ***Properties***. Если оно закрыто, то его можно включить командой ***View - Properties***. В этом окне отображаются важнейшие характеристики выделенного элемента.

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*** непосредственно без указания имени пространства, в котором они были определены.

Ключевое слово ***namespace*** создает для проекта свое собственное пространство имен, которое по умолчанию называется именем проекта. В нашем случае пространство имен называется Lab2.

C# - объектно-ориентированный язык, поэтому написанная на нем программа будет представлять собой совокупность взаимодействующих между собой классов. Автоматически был 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 языке C# должна иметь метод Main ().

Добавим в метод Main () следующий код:

```
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 метода имеет вид:

static void Main(string[] args)
{
объявление переменных;
ввод исходных данных;
расчет результата;
вывод результата;
}

Для хранения исходных данных и результатов надо выделить достаточно места в оперативной памяти. Для этого служит оператор 1. В нашей программе требуется хранить два значения: температуру по Цельсию и температуру по Фаренгейту, поэтому в операторе определяются две переменные. Одна, для хранения температуры по Фаренгейту, названа fahr, другая (по Цельсию) — cels. 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:

int – целочисленные,
float , **double** – вещественные
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 3 выполняется ввод с клавиатуры одного числа в переменную *fahr*.

ReadLine() – метод для ввода значения с клавиатуры. Данный метод возвращает введенную строку, поэтому с помощью метода Convert – преобразует данные в необходимый тип (например Convert.ToInt32() преобразует к целому типу, Convert.ToDouble() – к вещественному).

В операторе 4 вычисляется выражение, записанное справа от *операции присваивания* (обозначаемой знаком =), и результат присваивается переменной cels, то есть заносится в отведенную этой переменной память. Сначала **целая** константа 5 будет поделена на **целую** константу 9, затем результат этой операции умножен на результат вычитания числа 32 из переменной fahr.

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 ***Debug(Отладка)-Start Debugging(Начать отладку) или клавишу***  на панели инструментов

Как вы можете видеть, результат выполнения программы со стабильностью оказывается равным нулю! Это происходит из-за способа вычисления выражения. Давайте вновь обратимся к оператору 4. Константы 5 и 9 имеют целый тип, поэтому результат их деления также целочисленный. Естественно, что результат дальнейших вычислений не может быть ничем, кроме нуля. Исправить эту ошибку просто — достаточно записать хотя бы одну из констант в виде вещественного числа, например:

```
cels = 5.0 / 9 * (fahr - 32);
```

Задание 1

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

В языке C# предоставляется целый класс математических методов. Это класс - **Math**. Для вызова метода, необходимо прописать: *Math.Имя_метода();*

Обратите внимание на то, что вычисление синуса, косинуса и так далее вычисляется в радианах. Поэтому если вам нужны градусы, нужно конвертировать

```
int gradus = 30;
double radian = gradus * Math.PI / 180;
```

Таблица 2.1 – Математические методы C#

Метод	Назначение	Пример использования
Math.Abs	Возвращаем модуль числа.	`int i = Math.Abs(x);`
Math.Acos	Арккосинус. Определяется угол (в радианах) косинус которого равен указанному числу	`double x = Math.Acos(0.5);`
Math. Asin	Арксинус. Также определяет угол.	`double x = Math.Asin(0.5);`
Math.Atan	Арктангенс.	`double x = Math.Atan(O.S);`
Math .Cos	Возвращает косинус угла, заданного в радианах.	`double x = Math.Cos(1.04);`
Math .Exp	Экспонента.	`double x = Math.Exp(2);`
Math .Log	Вычисление логарифма. X - число которое нужно найти, Osn - основание логарифма.	`double x=Math.Log(X,Osn);`
Math .Log10	Вычисление десятичного логарифма.	`double x = Math.Log10(10)`
Math .Max	Возвращает из 2-х чисел большее число.	`int x =Math.Max(10,20);`
Math .Min	Возвращает из 2-х чисел меньшее число.	`intx = Math.Min( 10,20);`
Math .PI	Возвращает число Пи.	`double pi = Math.PI;`
Math.Pow	Вычисляет число возведенное в степень: a^x	`double i = Math.Pow(a, x);`
Math .Sin	Возвращает синус угла.	`double p = Math.Sln(0.5);`
Math.Sqrt	Возвращает квадратный корень.	`double r = Math.Sqrt(7);`
Math .Tan	Возвращает тангенс угла.	`double p = Math.Tan(1.4);`

По группам:

-константы E (число e) и PI (число пи)
- тригонометрические функции – Sin(), Cos(), Tan();
- обратные тригонометрические функции - ASin(), ACos(), ATan();
- гиперболические функции - Tanh(), Sinh(), Cosh();
- экспонента и логарифмические функции - Exp(), Log(), Log10();
- модуль, корень, степень - Abs(), Sqrt(), Pow;
- функции округления - Ceiling(), Floor(), Round();
- минимум, максимум, знак - Min(), Max(), Sign().

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

$z_2 = \mathrm{tg}3\alpha$

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

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

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

$z_2 = \sin\ \ y + x\ \cdot \sin\ \ y - x$

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

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

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

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

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

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

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

13) $z_1 = \frac{\sin\alpha + \cos\ 2\beta - \alpha}{\cos\alpha - \sin\ 2\beta - \alpha}$

$z_2 = \frac{1+\sin 2\beta}{\cos 2\beta}$

14) $z_1 = \frac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha}$ $z_2 = \mathrm{tg}2\alpha + \sec 2\alpha$

15) $z_1 = \frac{\sqrt{2b + 2\sqrt{b^2-4}}}{\sqrt{b^2-4} + b + 2}$ $z_2 = \frac{1}{\sqrt{b+2}}$

Лабораторная работа №3 «Организация ввода/вывода в консольных программах на языке C#»

Цель работы

– Освоение простейшей структуры программы на языке C#.

– Получение навыков в организации форматированного ввода-вывода на языке C#.

Постановка задачи

Написать программы, которые реализует диалог с пользователем. Вывод производить двумя способами: 1) слиянием строк; 2) форматированный вывод.

Общие сведения

Для работы с консолью используется стандартный класс Console, определенный в пространстве имен System.

Вывод данных. В приведенных выше примерах мы уже рассматривали метод WriteLine, который позволяет организовывать вывод данных на экран. Однако 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 последовательным 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*/
```

Замечание. Рассмотрим следующий фрагмент программы:

```
int i=3, j=4;
Console.WriteLine("{0} {1}", i, j);
```

При обращении к методу WriteLine через запятую перечисляются три аргумента: "{0} {1}", i, j. Первый аргумент определяет формат выходной строки. Следующие аргументы нумеруются с нуля, так переменная i имеет номер 0, j – номер 1. Значение переменной i будет помещено в выходную строку на место {0}, а значение переменной j –на место {1}. В результате на экран будет выведена строка: 3 4. Если запишем `Console.WriteLine("{0} {1} {2}", j, i, j)`, то на экран будет выведена строка: 4 3 4.

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), а k – количество позиций

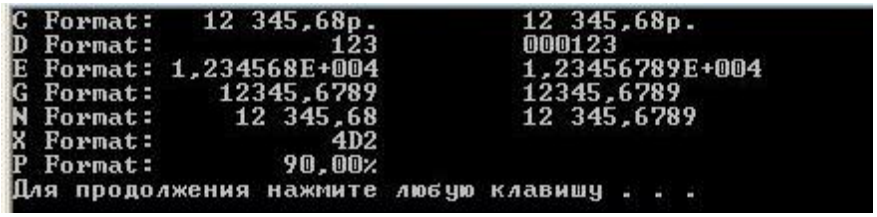
для дробной части значения идентификатора. В качестве спецификаторов могут использоваться следующие значения:

Таблица 3.2 – Форматы выводы

Параметр	Формат
C или c	Денежный. По умолчанию ставит знак р. k - задает количество десятичных разрядов
D или d	Целочисленный (используется только с целыми числами) k - задает минимальное количество цифр. При необходимости результат дополняется начальными нулями
E или e	Экспоненциальное представление чисел k - задает количество символов после запятой. По умолчанию используется 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

Пример:

```
static void Main()
   {
    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);       }
```



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). Пример:

```
static void Main()
 {  string s = Console.ReadLine();
  Console.WriteLine(s);
  }
```

Для того чтобы получить числовое значение необходимо воспользоваться преобразованием данных. Пример:

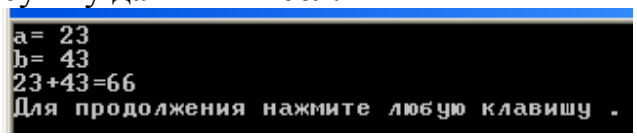
```
static void Main()
 {   int x = int.Parse(Console.ReadLine()); //преобразование введенной строки в число
```

```
Console.WriteLine(x); }
```

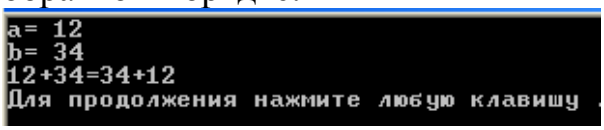
Для преобразования строкового представления целого числа в тип int мы используем метод int.Parse(), который реализован для всех числовых типов данных. Таким образом, если нам потребуется преобразовать строковое представление в вещественное используется float.Parse() или double.Parse().

Задание 1. Написать программу, которая, реализует диалог с пользователем:

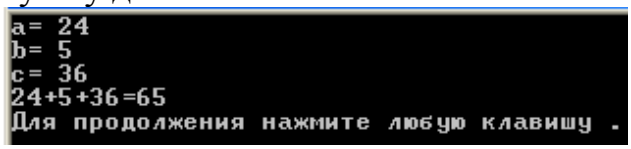
1) запрашивает с клавиатуры два целых числа, и выводит на экран сумму данных чисел:



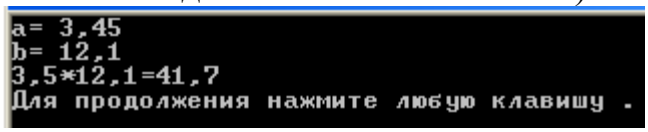
2) запрашивает с клавиатуры три целых числа, и выводит на экран сумму данных чисел в прямом и обратном порядке:



3) запрашивает с клавиатуры три целых числа, и выводит на экран сумму данных чисел:

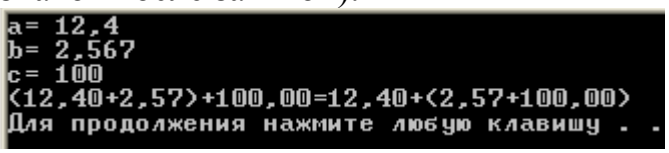


4) запрашивает с клавиатуры два вещественных числа, и выводит на экран произведение данных чисел (вещественные числа выводятся с точностью до 1 знака после запятой):

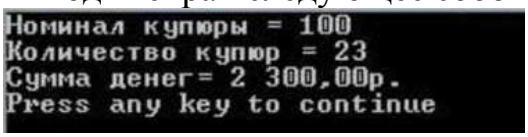


5) запрашивает с клавиатуры два вещественных числа, и выводит на экран результат деления первого числа на второе (вещественные числа выводятся с точностью до 3 знаков после запятой):

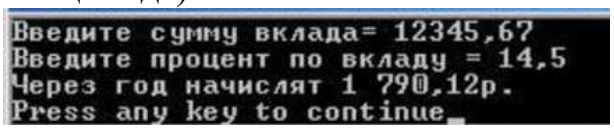
6) запрашивает с клавиатуры три вещественных числа, и выводит на следующее сообщение (вещественные числа выводятся с точностью до 2 знаков после запятой):



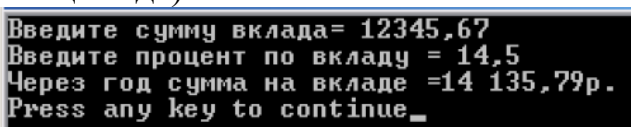
7) запрашивает с клавиатуры номинал купюры и количество купюр, и выводит экран следующее сообщение:



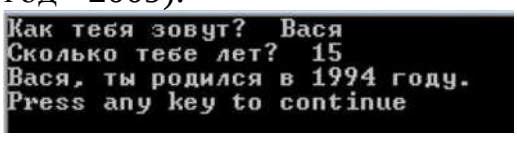
8) запрашивает с клавиатуры сумму вклада и процент по вкладу, и выводит на экран следующее сообщение (вклад без капитализации - все начисления в конце года):



9) запрашивает с клавиатуры сумму вклада и процент по вкладу, и выводит на экран следующее сообщение (вклад без капитализации - все начисления в конце года):



10) запрашивает с клавиатуры имя человека и его возраст, и выводит на экран следующее сообщение (текущий год - 2009):



Задание 2

1. Ввести с клавиатуры длины катетов a и b прямоугольного треугольника. Найти его периметр и площадь.
2. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
3. Найти длину окружности и площадь круга заданного радиуса R.
4. Найти площадь кольца, внутренний радиус которого равен R1, а внешний радиус равен R2 (R1 < R2). В качестве значения Pi использовать 3.14.
5. Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей.
6. Дана длина окружности. Найти площадь круга, ограниченного этой окружностью. В качестве значения Pi использовать 3.14.
7. Найти расстояние между двумя точками с заданными координатами (x1, y1) и (x2, y2).
8. Ввести с клавиатуры координаты трех вершин треугольника (x1, y1), (x2, y2), (x3, y3). Найти его периметр и площадь.
9. Найти действительные корни квадратного уравнения $A \cdot x^2 + B \cdot x + C = 0$, заданного своими коэффициентами A, B, C (коэффициент A не равен 0), если известно, что уравнение имеет ровно два корня.
10. Дано целое четырехзначное число. Используя операции деления / и нахождения остатка от деления %, найти сумму и произведение его цифр.
11. Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч (U < V). Время движения лодки по озеру T1 ч, а по реке (против течения) — T2 ч. Определить путь S, пройденный лодкой.
12. Скорость первого автомобиля V1 км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга.
13. Скорость первого автомобиля V1 км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили первоначально движутся навстречу друг другу.
14. Дана площадь круга. Найти длину окружности, ограничивающей этот круг. В качестве значения Pi использовать 3.14.
15. Найти периметр и площадь равнобедренной трапеции с основаниями a и b (a > b) и углом alpha при большем основании (угол дан в радианах).

Лабораторная работа № 4 Тема: «Программирование разветвляющихся алгоритмов»

Цель работы

- Закрепление структуры программы на языке C#
- Повторить базовые типы данных и математические функции
- Получение навыков в программирование разветвляющихся алгоритмов.

Постановка задачи

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

Общие сведения

Оператор if. Большинство операторов управления программой в любых языках программирования, включая C#, основываются на проверке условий, определяющих, какого рода действие необходимо выполнить. В результате проверки условий можно получить истину или ложь.

Стандартная форма записи оператора **if** следующая:

if *(выражение)*
оператор1;
else
оператор2;

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

Если выражение истинно (любое значение, кроме 0), выполняется блок операторов, следующий за **if**; иначе выполняется блок операторов, следующий за **else**.

Всегда выполняется код, ассоциированный или с **if**, или с **else**, но никогда не выполняются 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

Операнды в этих операциях должны быть арифметического типа или указателями.

Результат операции целочисленный: 0 (ложь) или 1 (истина).

Логические операции.

&& конъюнкция (И) арифметических операндов или отношений

|| дизъюнкция (ИЛИ) арифметических операндов или отношений

Результат 0 (ложь) или 1 (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. Типичной программистской конструкцией является *лесенка if-else-if* . Она выглядит следующим образом:

```
if (выражение)
  оператор;
else 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**.

Пример: Написать программу для вычисления значения функции

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

где a,b,c – целые числа, x – действительное число.

```
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 = 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();        }
```

Задание 1

1. Написать программу для вычисления значения функции

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

2. Написать программу для вычисления значения функции.

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

3. Написать программу для вычисления значения функции

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

4. Написать программу для вычисления значения функции

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

5. Написать программу для вычисления значения функции

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

6. Написать программу для вычисления значения функции

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

7. Написать программу для вычисления значения функции

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

8. Написать программу для вычисления значения функции

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

9. Написать программу для вычисления значения функции

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

10. Написать программу для вычисления значения функции.

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

Задание 2

1. Даны три целых числа. Найти количество положительных чисел в исходном наборе.
2. Даны три целых числа. Найти количество положительных и количество отрицательных чисел в исходном наборе.
3. Даны две переменные вещественного типа: A, B. Перераспределить значения данных переменных так, чтобы в A оказалось меньшее из значений, а в B — большее. Вывести новые значения переменных A и B.
4. Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной сумму этих значений, а если равны, то присвоить переменным нулевые значения. Вывести новые значения переменных A и B.
5. Даны две целые переменные: A и B. Если их значения не равны, то присвоить каждой переменной большее из этих значений, а если равны, то присвоить переменным нулевые значения. Вывести новые значения переменных A и B.
6. Даны три числа. Найти наименьшее из них.

7. Даны три числа. Найти среднее из них (то есть число, расположенное между наименьшим и наибольшим).
8. Даны три числа. Вывести вначале меньшее, а затем большее из данных чисел.
9. Даны три числа. Найти сумму двух наибольших из них.
10. Даны три переменные вещественного типа: A, B, C. Если их значения упорядочены по возрастанию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных A, B, C.
11. Даны три переменные вещественного типа: A, B, C. Если их значения упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. Вывести новые значения переменных A, B, C.
12. Даны три целых числа, одно из которых отлично от двух других, равных между собой. Определить порядковый номер числа, отличного от остальных.
13. Даны четыре целых числа, одно из которых отлично от трех других, равных между собой. Определить порядковый номер числа, отличного от остальных.
14. 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.
15. 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.

Лабораторная работа № 5 Тема «Программирование с использованием оператора switch»

Цель работы

Получение практических навыков в работе с оператором switch.

Постановка задачи

Напишите программу для решения задач.

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 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.

Пример: 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("понедельник"); break;
         case 2: Console.WriteLine("вторник"); break;
         case 3: Console.WriteLine("среда"); break;
         case 4: Console.WriteLine("четверг"); break;
         case 5: Console.WriteLine("пятница"); break;
         case 6: Console.WriteLine("суббота"); break;
         case 7: Console.WriteLine("воскресенье"); break;
         default: Console.WriteLine("ВЫ ОШИБЛИСЬ"); break;
  }    }
```

Задание 1

1. Дан пол человека: м – мужчина, ж – женщина. Вывести на экран возможные мужские и женские имена в зависимости от введенного пола.
2. Дан признак транспортного средства: а – автомобиль, в – велосипед, м - мотоцикл, с – самолет, п – поезд. Вывести на экран максимальную скорость транспортного средства в зависимости от введенного признака.
3. Дан номер телевизионного канала. Вывести на экран наиболее популярные программы заданного канала.
4. Дан признак геометрической фигуры на плоскости: к – круг, п – прямоугольник, т - треугольник. Вывести на экран периметр и площадь заданной фигуры (данные, необходимые для расчетов, запросить у пользователя).
5. Дан номер масти m ($1 \le m \le 4$), определить название масти. Масти нумеруются: «пики» - 1, «трефы» - 2, «бубны» - 3, «червы» - 4.
6. Дан номер карты k ($6 \le k \le 14$), определить достоинство карты. Достоинства определяются по следующему правилу: «туз» - 14, «король» - 13, «дама» - 12, «валет» - 11, «десятка» - 10, …, «шестерка» - 6.
7. Написать алгоритм, позволяющий получить словесное наименование школьных оценок.
8. Написать алгоритм, который по номеру дня недели - целому числу от 1 до 7 выдавать в качестве результата количество пар в Вашей группе в соответствующий день.
9. Написать алгоритм нахождения числа дней в месяце, если даны: Номер месяца n - целое число a, равное 1 для високосного года и равное 0 в противном случае.
10. В зависимости от того введена ли открытая скобка или закрытая, напечатать "открытая круглая скобка" или "закрытая фигурная скобка". (Учитывать круглые, квадратные, фигурные скобки).
11. В зависимости от введённого символа L, S, V программа должна вычислять длину окружности; площадь круга; объём цилиндра.

12. Напишите программу, которая по введенному числу из промежутка 1..12, определяет пору года.
13. Определить, является ли введенная буква русского алфавита гласной.
14. Напишите программу, которая по введенному числу из промежутка 0..24, определяет время суток.
15. Написать программу преобразования цифр в слова.

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.

Лабораторная работа № 6 Тема «Программирование циклических алгоритмов »

Цель работы

- Изучение циклических операторов языка 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.

Общие сведения

В C# и других современных 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, то есть тех операторов, которые выполняются несколько раз, начальных установок, модификации *параметра* цикла и проверки условия продолжения выполнения цикла. Один проход цикла называется итерацией.

Все циклы в C# выполняют тело цикла, пока условие истинно.

Стандартный вид цикла **for** следующий:

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

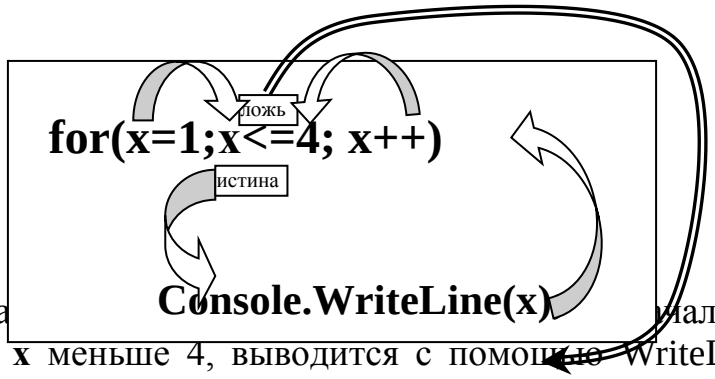
Оператор **for** имеет три главные части.

1. *Инициализация* – это выражение присваивания, используемое для установки начального значения переменной цикла.
2. *Условие* – это выражение, определяющее условие работы цикла.
3. *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.**

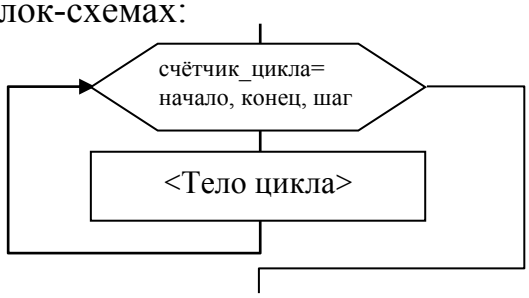
Порядок выполнения: переменной *счётчика цикла* присваивается начальное значение(инициализация) и проверяется условие; если условие неверно, то *тело цикла* не выполняется и управление передается на оператор, следующий за конструкцией **for**. Если же условие выполняется, то выполняется *тело цикла*, 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:



В да[illegible]ально 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. В данном примере **x** является *переменной цикла*, которая изменяется и проверяется на каждой итерации цикла.

Изображение в блок-схемах:



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.

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 равно 65. Обратим внимание, что в цикле переменная **x** *уменьшается*: сначала она

получает значение 100 и на каждой итерации цикла происходит уменьшение на 5.

Сокращенные операции в C#:

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

-- уменьшение на 1(a--)

Все составные операции присваивания представляют собой сокращенную запись простого присваивания для случая, который может быть описан как

X = XÅY; - тоже в записи составного присваивания X Å=Y.

В качестве символа Å могут быть использованы знаки операций: + - * / % (остаток от деления). Например, 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. Построить таблицу значений для функции

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

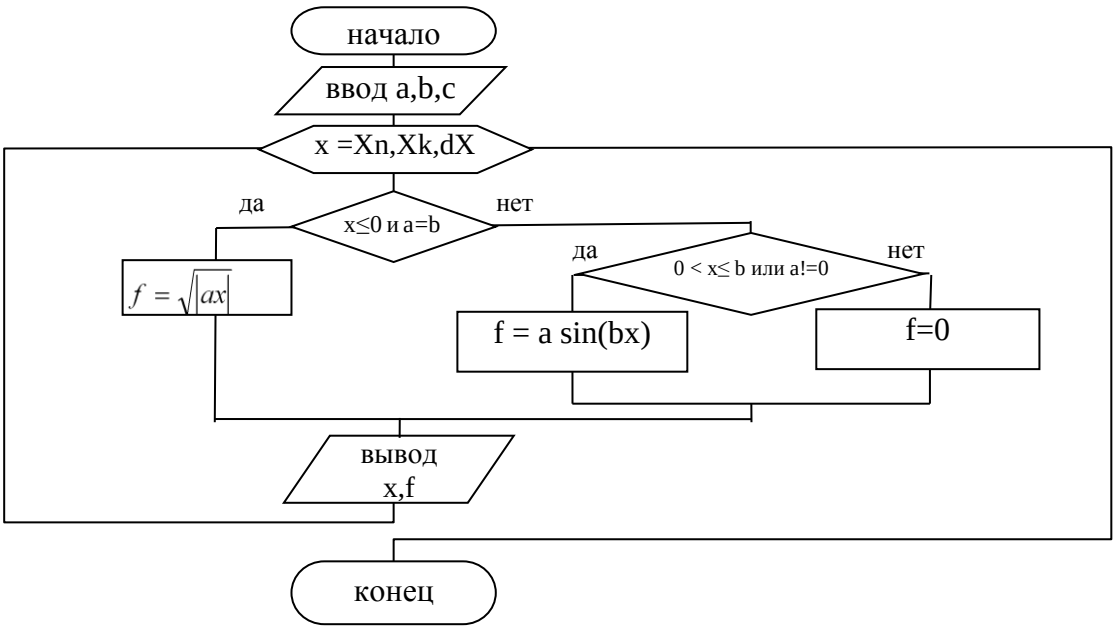
Значения a, b, $X_{нач}$, $X_{кон}$, dX ввести с клавиатуры.

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 имеет вид:



Задание 1. Циклический вычислительный процесс табулирование функции

1. Построить таблицу значений для функции $f(x) = x - \sin(x)$ на отрезке [0,π/2] с числом разбиений отрезка m=10.
2. Построить таблицу значений для функции $f(x) = \sin(x)$ на отрезке [π/4, π/2] с числом разбиений отрезка m=15.
3. Построить таблицу значений для функции $f(x) = \cos(x)$ на отрезке [π/3,2π/3] с числом разбиений отрезка m=20.
4. Построить таблицу значений для функции $f(x) = tg(x)$ на отрезке [0, π/4] с числом разбиений отрезка m=10.
5. Построить таблицу значений для функции $f(x) = ctg(x)$ на отрезке [π/4,π/2] с числом разбиений отрезка m=15.
6. Построить таблицу значений для функции $f(x) = \arcsin(x)$ на отрезке [0,1] с числом разбиений отрезка m=20.
7. Построить таблицу значений для функции $f(x) = \arccos(x)$ на отрезке [0.5,1] с числом разбиений отрезка m=10.
8. Построить таблицу значений для функции $f(x) = arctg(x)$ на отрезке [2,7] с числом разбиений отрезка m=15.
9. Построить таблицу значений для функции $f(x) = \sin(x) - \cos(x)$ на отрезке [0,π/2] с числом разбиений отрезка m=20.
10. Построить таблицу значений для функции $f(x) = x\sin(x)$ на отрезке [0,3π] с числом разбиений отрезка m=10.
11. Построить таблицу значений для функции $f(x) = \sin\left(\frac{1}{x}\right)$ на отрезке [π/8,2/π] с числом разбиений отрезка m=10.

Задание 2. Циклический вычислительный процесс конечные суммы и произведения.

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

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

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

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

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

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

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

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

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

12. Вычислить значение конечного произведения: $\frac{2}{\sqrt{2}}\cdot\frac{2}{\sqrt{2+\sqrt{2}}}\cdot\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}}\cdots\frac{2}{\underbrace{\sqrt{2+\cdots+\sqrt{2}}}_{n\text{ корней}}}$.

13. Вычислить значение конечного произведения: $(1+x)(1+x^2)\cdots(1+x^{2n})$.

14. Вычислить значение конечного произведения: $\left(1-\frac{2}{2(2+1)}\right)\left(1-\frac{2}{3(3+1)}\right)\cdots\left(1-\frac{2}{n(n+1)}\right)$.

15. Вычислить значение конечного произведения: $\frac{7}{9}\cdot\frac{26}{28}\cdots\frac{n^3-1}{n^3+1}$.

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

Цель работы

- Изучение циклических операторов языка 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.

Общие сведения

В C# и других современных языках программирования циклы позволяют выполнять набор операторов, пока выполняется некоторое условие.

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.

Перед тем как приступить к выполнению цикла, устанавливается начальное значение параметра цикла i, равное нулю. После выполнения тела цикла необходимо самостоятельно изменять значение параметра цикла, увеличивая его на единицу. Цикл будет прерван, как только i превысит 10.

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. Работа оператора **continue** чем-то похожа на работу оператора **break**. Но вместо форсированного окончания **continue** переходит к следующей итерации цикла, пропуская оставшийся код цикла. Например, следующая процедура выводит только положительные числа:

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

Пример. Найти сумму цифр введенного целого числа.

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

Для выделения цифр в числе воспользуемся следующим алгоритмом:

1 остаток от деления на 10 дает последнюю цифру числа;

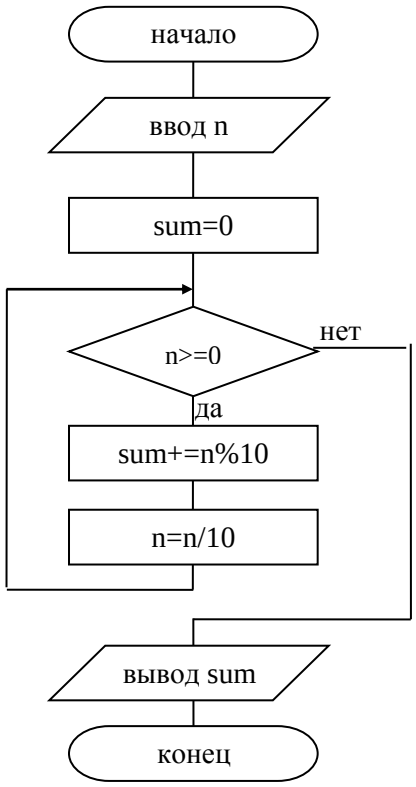
2 при делении на 10 последняя цифра числа отбрасывается;

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.

Текст программы:

```
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();
  }
```

Схема алгоритма программы:



Задание 1.

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

2) Вводить последовательность до тех пор, пока не встретятся 3 подряд идущих положительных числа. Тогда прервать ввод и сообщить, сколько во введенной последовательности было:

а) всего чисел,

б) положительных чисел,

в) отрицательных чисел.

3) Вычислить приближенное значение бесконечной суммы с точностью до ε=0,0001.

$$1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^n}+\cdots$$

4) Вычислить приближенное значение бесконечной суммы с точностью до ε=0,05.

$$1-\frac{1}{2}+\frac{1}{4}-\frac{1}{8}+\cdots\pm\frac{1}{2^n}\mp\cdots$$

5) Вычислить приближенное значение бесконечной суммы с точностью до ε=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.

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

Лабораторная работа № 8 Тема «Программирование циклических алгоритмов. Вложенные циклы.»

Цель работы

- Повторение циклических операторов языка C#.
- Получение навыков в программирование циклических алгоритмов.
- Изучение операторов перехода.

Постановка задачи

Разработать программы вложенных циклических процессов:

- вывода последовательности чисел в заданном виде;
- вычисление значений функции, путем разложения в ряд.

Общие сведения

Пример : Вывести на экран числа следующим образом:

1)
```
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);
} } }
```

Пример нахождения рекуррентной формулы

$$e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-\frac{x^2}{1!}+\frac{x^4}{2!}-\frac{x^6}{3!}+\ldots \quad |x|<\infty$$

Необходимо найти такой множитель, что зная предыдущее значение можно найти последующее,

т.е $\mathbf{a_n=T*a_{n-1}}$

Т.к.

$a_n = \frac{(-1)^n x^{2n}}{n!}$; $a_{n-1} = \frac{(-1)^{n-1} x^{2n-2}}{(n-1)!}$ то $T = \frac{a_n}{a_{n-1}} = \frac{\frac{(-1)^n x^{2n}}{n!}}{\frac{(-1)^{n-1} x^{2n-2}}{(n-1)!}} = \frac{(-1)x^2}{n}$

Действительно, из формулы видим $a_0=1$

Находим a_1(n=1): $a_1 = T \cdot a_0 = \frac{(-1)x^2}{n} \cdot a_0 = -\frac{x^2}{1}$

Находим a_2(n=2): $a_2 = T \cdot a_1 = \frac{(-1)x^2}{n} \cdot a_1 = -\frac{x^2}{2} \cdot (-\frac{x^2}{1}) = \frac{x^4}{2!}$

Задание 1.

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

2)
5
5 5
5 5 5
5 5 5 5
5 5 5 5 5

3)
1 1 1 1 1
1 1 1 1
1 1 1
1 1
1

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

5)
6 6 6 6 6
7 7 7 7
8 8 8
9 9
10

6)
7
6 6
5 5 5
4 4 4 4
3 3 3 3 3

7)
8 8 8 8 8
7 7 7 7
6 6 6
5 5
4

8)
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

9)
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1

10)
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0

11)
4 3 2 1 0
3 2 1 0
2 1 0
1 0
0

12)
1
0
2 2
0 0
3 3 3
0 0 0
4 4 4 4
0 0 0 0
5 5 5 5 5
0 0 0 0 0

13)
8
7
7 7
6 6
6 6 6
5 5 5
5 5 5 5
4 4 4 4

14)
1
6
2 2
7 7
3 3 3
8 8 8
4 4 4 4
9 9 9 9

15)
9
4
8 8
3 3
7 7 7
2 2 2 2
6 6 6 6 6
1 1 1 1 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. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество

просуммированных членов ряда. Для вычисления последующего члена ряда использовать рекуррентную формулу.

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

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

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

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

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

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

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

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

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

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

12 $$\arcsin x = x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \ldots \cdot 2n \cdot (2n+1)} = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} - \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \cdots \quad |x| < 1$$

13 $arctg\, x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \cdots, \quad x < -1$

14 $e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - \frac{x^2}{1!} + \frac{x^4}{2!} - \frac{x^6}{3!} + \ldots \quad |x| < \infty$

15

$$\arccos x = \frac{\pi}{2} - (x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \ldots \cdot 2n \cdot (2n+1)}) =$$

$$= \frac{\pi}{2} - (x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} - \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \cdots), \quad |x| < 1$$

Лабораторная работа № 9 Тема: «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. Написать программу, которая работает с любым набором данных.

Общие сведения

Массив – это совокупность переменных одного типа, к которым обращаются с помощью общего имени. Доступ к отдельному элементу массива может осуществляться с помощью индекса. В языке C# все массивы состоят из соприкасающихся участков памяти. Наименьший адрес соответствует первому элементу, наибольший адрес соответствует последнему элементу. Массивы могут иметь одну или несколько размерностей.

Одномерные массивы

Одномерный массив – это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в C# начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Создание массива представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяют размер массива.

В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

Форма записи	***Пояснения***
базовый_тип [] имя_массива; *Например*: `int [] a;`	Описана ссылка на одномерный массив, которая в дальнейшем может быть использована: 1) для адресации на уже существующий массив; 2) передачи массива в метод в качестве параметра 3) отсроченного выделения памяти под элементы массива.
базовый_тип [] имя_массива =	Объявлен одномерный массив заданного типа и выделена

new базовый_тип [размер]; *Например*: `int []a=new int [10];`	память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. Элементы массива равны нулю. 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 происходи с помощью индекса, для этого нужно указать имя массива и в квадратных скобках его номер. Например, a[0], b[10], c[i].

***Замечание**. В C# индексация массивов начинается с нуля.*

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.

Пример 1.

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

Пример 2.

```
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]);
```

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:

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

б) 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 так далее. Тем самым самое наибольшее или наименьшее число передвинется на последнее место. Следующие просмотры следует начинать опять сначала, уменьшая на единицу количество просматриваемых элементов. Массив будет упорядочен после просмотра, в котором участвовали только первый и второй элементы. (Сортировка обменами)

в) 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)

г) 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)

Задание

1(10). В одномерном массиве, состоящем из n вещественных элементов, вычислить: 1) сумму отрицательных элементов массива;

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

2(9). В одномерном массиве, состоящем из n вещественных элементов, вычислить: 1) сумму положительных элементов массива;

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

3(8). В одномерном массиве, состоящем из n целых элементов, вычислить:

1) произведение элементов массива с четными номерами;

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

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом — все отрицательные (элементы, равные 0, считать положительными).

4(7). В одномерном массиве, состоящем из n вещественных элементов, вычислить: 1) сумму элементов массива с нечетными номерами;

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

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

5(6). В одномерном массиве, состоящем из n вещественных элементов, вычислить: 1) максимальный элемент массива;

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

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

6(5). В одномерном массиве, состоящем из n вещественных элементов, вычислить: 1) минимальный элемент массива;

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

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

7(4). В одномерном массиве, состоящем из *n* целых элементов, вычислить:

1) номер максимального элемента массива;

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

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

8(3). В одномерном массиве, состоящем из *n* вещественных элементов, вычислить: 1) номер минимального элемента массива;

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

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

9(2). В одномерном массиве, состоящем из *n* вещественных элементов, вычислить: 1) максимальный по модулю элемент массива;

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

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

10(1). В одномерном массиве, состоящем из *n* целых элементов, вычислить: 1) минимальный по модулю элемент массива;

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

11(15). В одномерном массиве, состоящем из n вещественных элементов, вычислить: 1) номер минимального по модулю элемента массива;

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

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

12(13). В одномерном массиве, состоящем из *n* вещественных элементов, вычислить: 1) номер максимального по модулю элемента массива;

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

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

13(14). В одномерном массиве, состоящем из *n* вещественных элементов, вычислить: 1) количество элементов массива, лежащих в диапазоне от A до B;

2) сумму элементов массива, расположенных после максимального элемента.

Упорядочить элементы массива по убыванию модулей элементов.

14(12). В одномерном массиве, состоящем из *n* вещественных элементов, вычислить: 1) количество элементов массива, равных 0;

2) сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей элементов.

15(11). В одномерном массиве, состоящем из *n* вещественных элементов, вычислить: 1) количество элементов массива, больших С;

2) произведение элементов массива, расположенных после максимального по модулю элемента.

Преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом — все положительные (элементы, равные 0, считать положительными).

Лабораторная работа № 10 Тема: «Обработка массивов как объектов»

Цель работы

- Получение практических навыков в работе с одномерными массивами.
- Знакомство с классом Array.

Постановка задачи

Для конкретного варианта (задания из лабораторной работы №9, номер варианта для лабораторной работы №10 указан в скобках) задать массив исходных данных случайными числами и выполнить над ним указанные действия с использованием класса Array.

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. Например, свойство Length позволяет определять количество элементов в массиве:

```
 static void Main()
{ int[] myArray = { 0, -1, -2, 3, 4, 5, -6, -7, 8, -9 };
  for (int i = 0; i < myArray.Length; i++) Console.Write("{0} ", myArray [i]);
  Console.WriteLine();         }
```

Другие свойства и методы класса Array приведены в следующей таблице:

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 массив
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). В данном случае мы обращаемся к статическому методу Sort класса Array и передаем данному методу в качестве параметра объект myArray - экземпляр класса Array.

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
            {   int[] MyArray;
                Console.Write("Введите размерность массива: ");
                int n = int.Parse(Console.ReadLine());
                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();
        }
```

Задание. Варианты задания из лабораторной работы №9, номер варианта для лабораторной работы №10 указан в скобках

Лабораторная работа № 11 Тема: «Двумерные массивы»

Цель работы

Целью лабораторной работы является:

- Изучение двумерных и ступенчатых массивов;
- закрепление навыков структурного программирования.

Постановка задачи

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

Основные сведения

Многомерные массивы

Разделение *массивов* на одномерные и *многомерные* носит исторический характер. Никакой принципиальной разницы между ними нет. Одномерные *массивы* -это частный случай *многомерных*. Можно говорить и по-другому: *многомерные массивы* являются естественным обобщением одномерных. Одномерные *массивы* позволяют задавать такие математические структуры как векторы, двумерные - матрицы, трехмерные - кубы данных, *массивы* большей *размерности* - многомерные кубы данных.

В чем особенность объявления *многомерного массива*? Как в типе указать *размерность массива*? Это делается достаточно просто, за счет использования запятых. Вот как выглядит объявление *многомерного массива* в общем случае:

<тип>[, ... ,] <объявители>;

Число запятых, увеличенное на единицу, и задает *размерность массива*. Что касается объявителей, то все, что сказано для одномерных *массивов*, справедливо и для *многомерных*. Можно лишь отметить, что хотя явная инициализация с использованием *многомерных* константных *массивов* возможна, но применяется редко из-за громоздкости такой структуры. Проще инициализацию реализовать программно, но иногда она все же применяется. Вот пример:

```
int[,] matrix = { { 1, 2 }, { 3, 4 } };
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();
}
```

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.

Пример:

```
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[][].

А вот конструкторы нижнего уровня необходимы. Еще одно важное замечание - *динамические массивы* возможны и здесь. В общем случае, границы на любом уровне могут быть выражениями, зависящими от переменных. Более того, допустимо, чтобы *массивы* на нижнем уровне были *многомерными*

Задания

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 x 4. Написать программу для вычисления I нормы матрицы: $\|A\|_I = \max\limits_{i=1,n} \sum\limits_{k=1}^{n} \left|a_{ik}\right|$.
6. Найти наибольший элемент матрицы A(5x3) и номер строки и столбца, в котором он находится.
7. Вычислить сумму элементов каждой строки матрицы X(4x3), определить наименьшее значение этих сумм и номер соответствующей строки.
8. Найти наибольшие элементы каждой строки матрицы X(4x5*)* и записать их в массив Y.
9. Найти наибольший элемент главной диагонали матрицы A(4x4*)* 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.

Примечание. Матрица A имеет седловую точку A_{ij}, если A_{ij} является минимальным элементом в i-й строке и максимальным в j-ом 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 $i\text{-}1 \le k \le i\text{+}1$, $j\text{-}1 \le l \le j\text{+}1$, $(k, l) \ne (i, j)$. Операция сглаживания матрицы дает новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы

размером 10×10. В сглаженной матрице найти сумму модулей элементов, расположенных ниже главной диагонали.

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

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

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

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

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

15. Дана целочисленная прямоугольная матрица. Определить номер первого из столбцов, содержащих хотя бы один нулевой элемент. Характеристикой строки целочисленной матрицы назовем сумму ее отрицательных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.

Лабораторная работа № 12 Тема: «Символьные и строковые типы»

Цель работы

- Изучение строковых типов;
- использование строковых типов.

Постановка задачи

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

Основные сведения

Класс char. В C# есть *символьный класс* ***char***, основанный на классе System.Char и использующий двухбайтную кодировку Unicode представления символов.

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	Все символы разделены на категории. 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. Естественно, 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

Класс string. Основным типом при работе со строками является тип *string*, задающий строки переменной длины. Объекты *класса string* объявляются как все прочие объекты простых типов - с явной или отложенной инициализацией, с явным или неявным вызовом конструктора класса. У класса *string* достаточно много конструкторов. Они позволяют сконструировать строку из:

- символа, повторенного заданное число раз;
- массива символов char[];
- части массива символов.

Примеры объявления строк с вызовом разных конструкторов:

```
string world = "Мир";
string sss = new string('s', 5);
char[] yes = "Yes".ToCharArray();
string syes = new string(yes);
string sye = new string(yes, 0, 2);
Console.WriteLine("world = {0}; sss={1}; syes={2}; sye= {3}", world, sss, syes, sye);
```

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

Операции над строками.Над строками определены следующие операции:

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

Бинарная операция "+" сцепляет две строки, приписывая вторую строку к хвосту первой.

Взятие индекса позволяет рассматривать строку как массив и получать без труда каждый ее символ. Каждый символ строки имеет тип char, доступный только для чтения, но не для записи.

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);
s2 = s1;
b1 = (s1 != s2);
ch2 = s2[0];
Console.WriteLine("s1={0}, s2={1}, b1={2}, ch1={3}, ch2={4}", s1,s2,b1,ch1,ch2);
```

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	Выполняет форматирование в соответствии с заданными спецификациями формата. Ниже приведено более полное описание метода

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.

Рассмотрим примеры применения этих методов. В первом из них строка представляет сложноподчиненное предложение, которое разбивается на простые предложения. Во втором предложение разделяется на слова. Затем производится обратная сборка разобранного текста. Вот 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.

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	Выполняет набивку нужным числом 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

Задание 1.

1. Для заданной строки символов проверить, является ли она симметричной или нет. (Симметричной считается строка, которая одинаково читается слева направо и справа налево).
2. Для заданной строки символов определить сумму всех входящих в неё цифр.
3. Для заданной строки определить все входящие в неё символы. Например: строка "abccbbbabba" состоит из символов "a","b" и "c".
4. Задана строка символов. Определить, какой символ встречается в этой строке подряд наибольшее число раз. В ответе указать символ, образующий самую длинную последовательность, длину последовательности.
5. Для заданной строки символов, состоящей из строчных букв и пробелов, определить слово наибольшей длины, которое начинается и заканчивается на одну и ту же букву.
6. Задана строка символов, содержащая два или более слов, разделенных пробелами. Написать программу, меняющую местами все четные и нечетные слова в строке.
7. Подсчитать, сколько раз в данной строке встречается некоторая буква, вводимая с клавиатуры.
8. Из строки удалить среднюю букву, если длина строки нечетная, если четная — удалить две средние буквы.
9. Заменить все вхождения в текст некоторой буквы на другую букву (их значения вводить с клавиатуры).
10. Заменить все вхождения подстроки Str1 на подстроку Str2 (подстроки вводятся с клавиатуры)
11. Дана последовательность слов. Напечатать все слова в алфавитном порядке.

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

Задание 2.

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>0 и символ C. Вывести строку длины N, которая состоит из символов C.
9. Дано четное число N>0 и символы C1 и C2. Вывести строку длины N, которая состоит из чередующихся символов C1 и C2, начиная с C1.
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>0. Вывести строку, содержащую символы строки S, между которыми вставлено по N символов «*» (звездочка).
13. Дана строка. Подсчитать количество содержащихся в ней цифр.
14. Дана строка. Подсчитать количество содержащихся в ней прописных латинских букв.
15. Дана строка. Подсчитать общее количество содержащихся в ней строчных латинских и русских букв.

Задание 3. *Дана строка, в которой содержится осмысленное текстовое сообщение. Слова сообщения разделяются пробелами и знаками препинания.*

Вывести только те слова сообщения, в которых содержится заданная подстрока.

Пример

```
Console.WriteLine("Введите строку: ");
StringBuilder a = new StringBuilder(Console.ReadLine());
Console.WriteLine("Исходная строка: "+a);
Console.WriteLine("Введите заданные подстроку: ");
string x=Console.ReadLine();
for (int i=0; i<a.Length;)
      if (char.IsPunctuation(a[i]))a.Remove(i,1);
      else ++i;
string str=a.ToString();
str=str.Trim();
string []s=str.Split(' ');
```

```
Console.WriteLine("Искомые слова: ");
for (int i=0; i<s.Length; ++i)
      if (s[i].IndexOf(x)!=-1) Console.WriteLine(s[i]);
```

1. Вывести только те слова сообщения, которые содержат не более чем n букв.
2. Вывести только те слова сообщения, которые начинаются с прописной буквы.
3. Вывести только те слова сообщения, которые содержат хотя бы одну цифру.
4. Удалить из сообщения все слова, которые заканчиваются на заданный символ.
5. Удалить из сообщения все слова, содержащие данный символ (без учета регистра).
6. Удалить из сообщения все однобуквенные слова (вместе с лишними пробелами).
7. Удалить из сообщения все повторяющиеся слова (без учета регистра).
8. Подсчитать сколько раз заданное слово встречается в сообщении.
9. Подсчитать сколько слов, состоящих только из прописных букв, содержится в сообщении.
10. Найти самое длинное слово сообщения.
11. Найти все самые длинные слова сообщения.
12. Найти самое короткое слово сообщения.
13. Найти все самые короткие слова сообщения.
14. Вывести на экран все слова-палиндромы, содержащиеся в сообщении.
15. Вывести только те слова, которые встречаются в тексте ровно один раз.

Задание 4.

1. Даны строки S, S1 и S2. Заменить в строке S последнее вхождение строки S1 на строку S2.
2. Даны строки S, S1 и S2. Заменить в строке S все вхождения строки S1 на строку S2.
3. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и вторым пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.
4. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и последним пробелом.
5. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти количество слов в строке.
6. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые начинаются и заканчиваются одной и той же буквой.
7. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые содержат хотя бы одну букву «А».
8. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые содержат ровно три буквы «А».

9. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого короткого слова.
10. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого длинного слова.
11. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним символом «.» (точка). В конце строки точку не ставить.
12. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, заменив в нем все последующие вхождения его первой буквы на символ «.» (точка). Например, слово «МИНИМУМ» надо преобразовать в «МИНИ.У.». Количество пробелов между словами не изменять.
13. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в обратном порядке.
14. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в алфавитном порядке.
15. Дана строка-предложение на русском языке. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы. Словом считать набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки. Слова, не начинающиеся с буквы, не изменять.

Список литературы

1 **Павловская, Т. А.** C#. Программирование на языке высокого уровня: учебник для вузов / Т. А. Павловская. – Санкт-Петербург.: Питер, 2010. – 432 с.

2 **Фаронов, В. В.** Программирование на языке C# / В. В. Фаронов. – Санкт-Петербург: Питер, 2007. – 240 с.

3 **Шилдт, Г.** C# 2.0. Полное руководство: пер. с англ. / Г. Шилдт – Москва: ЭКОМ, 2007. – 976 с.

4 **Казаков, П. В.** Языки программирования: лабораторный практикум для вузов / П. В. Казаков. – Брянск: БГТУ, 2011. – 84 с.

5 C# 2005 для профессионалов: пер. с англ. / К. Нейгел [и др.] – Москва: Диалектика, 2007. – 1376 с.

6 **Ишкова, Э. А.** C#. Начала программирования: учебник / Э. А. Ишкова. – Москва: Бином-Пресс, 2010. – 336 с.