

ПЕРМСКИЙ
ГОСУДАРСТВЕННЫЙ
НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Т. В. Ромашкина, Н. И. Миндоров

АЛГОРИТМИЗАЦИЯ И ОСНОВЫ ПРОГРАММИРОВАНИЯ

ОСНОВЫ ОБЪЕКТНО-
ОРИЕНТИРОВАННОГО
ПРОГРАММИРОВАНИЯ. C#



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

Т. В. Ромашкина, Н. И. Миндоров

АЛГОРИТМИЗАЦИЯ И ОСНОВЫ ПРОГРАММИРОВАНИЯ ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ. C#

*Допущено методическим советом
Пермского государственного национального
исследовательского университета в качестве
учебного пособия для студентов, обучающихся
по направлениям подготовки бакалавров «Математика»,
«Механика и математическое моделирование»
и по специальности «Фундаментальная математика и механика»*



Пермь 2021

УДК 004.4(075.8)
ББК 22.2
Р698

Ромашкина Т. В.

Р698 Алгоритмизация и основы программирования. Основы объектно-ориентированного программирования. С# [Электронный ресурс] : учебное пособие / Т. В. Ромашкина, Н. И. Миндоров ; Пермский государственный национальный исследовательский университет. – Электронные данные. – Пермь, 2021. – 5,29 Мб ; 159 с. – Режим доступа: <http://www.psu.ru/files/docs/science/books/uchebnie-posobiya/romashkina-mindorov-algoritmizaciya-i-osnovy-programmirovaniya.pdf>. – Заглавие с экрана.

ISBN 978-5-7944-3682-2

Учебное пособие включает в себя материал для изучения основ объектно-ориентированного программирования с использованием языка С#. Издание содержит синтаксические конструкции, операторы управления, объектную модель, используемые в рассматриваемом языке программирования. Для каждого раздела приведены примеры решения различных задач с кодом на языке программирования и комментариями к нему.

Пособие предназначено для студентов первого курса вуза, изучающих дисциплину «Алгоритмизация и основы программирования».

УДК 004.4(075.8)
ББК 22.2

*Издается по решению ученого совета механико-математического факультета
Пермского государственного национального исследовательского университета*

Рецензенты: кафедра информатики и вычислительной техники Пермского государственного гуманитарно-педагогического университета (и.о. зав. кафедрой – канд. техн. наук, доцент **И. П. Половина**);
доцент кафедры информационных технологий в бизнесе НИУ ВШЭ-Пермь, канд. физ.-мат. наук, доцент **Л. В. Шестакова**

ISBN 978-5-7944-3682-2

© ПГНИУ, 2021
© Ромашкина Т. В., Миндоров Н. И., 2021

ВВЕДЕНИЕ	5
ОРГАНИЗАЦИЯ ПРОГРАММ ЛИНЕЙНОЙ СТРУКТУРЫ	6
ОБЩАЯ ХАРАКТЕРИСТИКА ЭЛЕМЕНТОВ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#	6
Задания для выполнения в аудитории	11
Задания для самостоятельной работы студента (СРС)	14
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	15
Приложение 1	15
Приложение 2	16
Приложение 3	17
Приложение 4	18
Приложение 5	19
Приложение 6	21
ОПЕРАТОРЫ ВЫБОРА	22
ОПЕРАТОР "if"	22
ОПЕРАТОР "?"	23
ОПЕРАТОР "switch case"	24
Задания для выполнения в аудитории	27
Задания для самостоятельной работы студента (СРС)	28
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	29
ОПЕРАТОРЫ ЦИКЛА	30
ЦИКЛЫ: "while" И "do - while "	30
Задания для выполнения в аудитории	33
Задания для самостоятельной работы студента (СРС)	34
ЦИКЛ "for"	36
Задания для выполнения в аудитории	38
Задания для самостоятельной работы студента (СРС)	39
ВЛОЖЕННЫЕ ЦИКЛЫ	41
Задания для выполнения в аудитории	43
Задания для самостоятельной работы студента (СРС)	43
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	44
РЕКУРРЕНТНЫЕ СООТНОШЕНИЯ	45
ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ С ПОМОЩЬЮ РЕКУРРЕНТНЫХ СООТНОШЕНИЙ	45
Задания для выполнения в аудитории	49
Задания для самостоятельной работы студента (СРС)	51
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	52
МЕТОДЫ	53
СТАТИЧЕСКИЕ МЕТОДЫ	54
Задания для выполнения в аудитории	58
Задания для самостоятельной работы студента (СРС)	59
РЕКУРСИВНЫЕ МЕТОДЫ	60
Задания для выполнения в аудитории	63
Задания для самостоятельной работы студента (СРС)	64
НЕСТАТИЧЕСКИЕ МЕТОДЫ	66
Задания для выполнения в аудитории	70
Задания для самостоятельной работы студента (СРС)	71
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	74
Приложение 1	75
КОНСТРУКТОРЫ	76
Задания для выполнения в аудитории	83
Задания для самостоятельной работы студента (СРС)	84

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	85
СВОЙСТВА	86
Задания для выполнения в аудитории	89
Задания для самостоятельной работы студента (СРС)	90
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	92
МАССИВЫ	93
ОДНОМЕРНЫЕ МАССИВЫ	93
СПОСОБЫ ОРГАНИЗАЦИИ МАССИВА	93
ДЕЙСТВИЯ НАД ЭЛЕМЕНТАМИ МАССИВА	95
Задания для выполнения в аудитории	102
Задания для самостоятельной работы студента (СРС)	103
ВСТАВКА И УДАЛЕНИЕ ЭЛЕМЕНТОВ МАССИВА	108
Задания для выполнения в аудитории	110
Задания для самостоятельной работы студента (СРС)	114
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	120
Приложение 1	120
Приложение 2	121
СОРТИРОВКИ ПРОСТЫЕ	122
Задания для выполнения в аудитории	126
Задания для самостоятельной работы студента (СРС)	127
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	128
Приложение 1	129
Приложение 2	132
СОРТИРОВКИ БЫСТРЫЕ	134
СОРТИРОВКА СЛИЯНИЕМ	134
Задания для выполнения в аудитории	140
Задания для самостоятельной работы студента (СРС)	140
СОРТИРОВКА ХОАРА (Быстрая сортировка)	141
Задания для выполнения в аудитории	144
Задания для самостоятельной работы студента (СРС)	144
МНОГОМЕРНЫЕ МАССИВЫ	145
ДВУМЕРНЫЕ МАССИВЫ	145
СПОСОБЫ ОРГАНИЗАЦИИ МАССИВА	145
ДЕЙСТВИЯ НАД ЭЛЕМЕНТАМИ МАССИВА	146
Задания для выполнения в аудитории	151
Задания для самостоятельной работы студента (СРС)	151
ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ	155
Приложение 1	156
Приложение 2	157

ВВЕДЕНИЕ

Учебное пособие разработано в соответствии с программой дисциплины «Алгоритмизация и основы программирования» и предназначено для изучения указанной дисциплины студентами I курса направлений «Математика», «Механика и математическое моделирование», а также специальности «Фундаментальные математика и механика».

Пособие состоит из двух частей.

В первой части рассматривались общие вопросы информатики, машинное представление чисел, измерение информации, алгоритмы кодирования информации, работа процессора.

Во второй части изложены основы объектно-ориентированного программирования. На практических примерах рассмотрено программирование на языке C#, начиная с основ языка. Описаны синтаксические конструкции, операторы управления, объектная модель, используемые в языке программирования C#. В данное пособие включены темы, связанные с организацией вычислений с помощью рекуррентных соотношений; с изучением статических, рекурсивных, нестатических методов. Уделено внимание рассмотрению конструкторов как методов для инициализации объектов; свойств как методов доступа к полям класса; массивов одномерных и многомерных (двумерных).

Структура пособия представлена следующим образом:

- Теоретические сведения по изучаемым темам, с необходимыми примерами программного кода. Приведенные примеры имеют подробное описание и комментарии. По окончании рассмотрения примеров, предусмотрены задания по изученному материалу.
- Задания по изученному теоретическому материалу:
 - *Задания для выполнения в аудитории.* Выполнение данных заданий предполагается «в присутствии преподавателя». При необходимости у студента есть возможность получить консультацию или помощь от преподавателя.
 - *Задания для самостоятельной работы студента (СРС).* Данные задания предназначены как для закрепления изученного материала, так и для самостоятельного изучения некоторых тем дисциплины. Материал, предназначенный для самостоятельного освоения, снабжен пошаговыми инструкциями (алгоритмом работы). Выполнение данных заданий предполагается «без участия преподавателя».
- Информационные источники по каждой теме.
- Приложения с необходимой справочной и дополнительной информацией по изучаемому материалу.

ОРГАНИЗАЦИЯ ПРОГРАММ ЛИНЕЙНОЙ СТРУКТУРЫ

ОБЩАЯ ХАРАКТЕРИСТИКА ЭЛЕМЕНТОВ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#

Знаки операций и разделители

Примеры разделителей: скобки, точка, запятая. Некоторые знаки операций и разделители:

{ } [] () + - * / =

< > ? ++ || «...»

+= -= *= /= %

<= >= == ! & | ||

Комментарии

Однострочный комментарий - (//)

Многострочный комментарий заключается между символами-скобками /* и */

Типы данных

Тип переменной указывает на то, какие данные могут быть сохранены в этом участке памяти и в каких действиях эта переменная может участвовать. В зависимости от признака классификации, типы разделяют на:

- простые и структурированные;
- статические и динамические;
- встроенные и определенные пользователем.

Встроенные (базовые) типы

К встроенным типам относятся:

- *тип целых чисел*

int -2147483648 до 2147483647

Int16 -32768..32768

Int32 -2млрд..2млрд

Int64 -9223372036854775808 до 9223372036854775807

Byte 0..255

- *тип действительных (вещественных) чисел*

double от $\pm 5,0 \times 10^{-324}$ до $\pm 1,7 \times 10^{308}$

float от $\pm 1,5 \times 10^{-45}$ до $\pm 3,4 \times 10^{38}$

Обозначения действительного числа:

-25.000452

0.24

4.854E-12

- *символьный тип*

char c1 = 'Z'; // Буквенный символ

char c2 = '\x0058'; // Шестнадцатеричный код символа

- *строковый тип*

string

Тип данных string — это последовательность, содержащая ни одного или любое число знаков Юникода. В платформе .NET Framework string является псевдонимом для String.

- *логический тип* – bool. Может принимать два значения Истинно-true Ложно-false, например: bool f = true.

Таблица встроенных типов C# представлена в Приложении 1.

Описание переменных

int a - описание целочисленной переменной "a"

double t - описание переменной "t" вещественного типа

float W - описание переменной "W" вещественного типа

string p - описание переменной "p" строкового типа

Арифметические операции

При вычислении выражения, стоящего в правой части оператора присвоения используются арифметические операции:

+ – сложение;

– – вычитание;

* – умножение;

/ – деление;

% – остаток от деления;

++ - инкремент (увеличение на 1);

-- декремент (уменьшение на 1).

Пример 1. Деление

Операция	Типы операндов		Действие	Типы результата
/	Целый	Целый	Деление целочисленное	Целый
/	Вещественный	Целый	Деление "обычное"	Вещественный
/	Целый	Вещественный	Деление "обычное"	Вещественный
/	Вещественный	Вещественный	Деление "обычное"	Вещественный
%	Целый	Целый	Остаток от деления	Целый
%	Вещественный	Вещественный	Остаток от деления	Вещественный

Выражение	Результат	Выражение	Результат
16/3	5	11 / 3,8	2,89473684210526
16 % 3	1	3,8 / 11	0,345454545454545
(double)16 / 3	5,33333333333333	3,8 / 3,7	1,02702702702703
16 / (double)3	5,33333333333333	3,7 % 3,8	3,7

Пример 2. Инкремент, декремент

Данные операции имеют *префиксную* (++a), и *постфиксную* (a++) формы записи.

Операции: ++a; --a изменяют значения переменной и запоминают результат в ту же переменную (в переменную a).

Операции: a++; a-- сохраняют исходное значение, затем возвращают его как результат.

++a равносильно записи: **a = a + 1**

--a равносильно записи: **a = a - 1**

Исходные данные	Выражение	Результат
a = 3; b = 4;	a++	a = 4; b = 4;
a = 3; b = 4;	a -- b ++	a = 2; b = 5;

Составной оператор присваивания

a + = a равносильно записи **a=a+a**

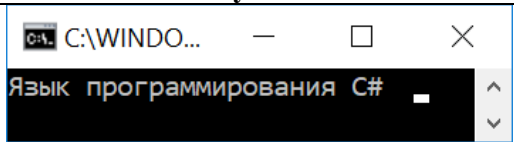
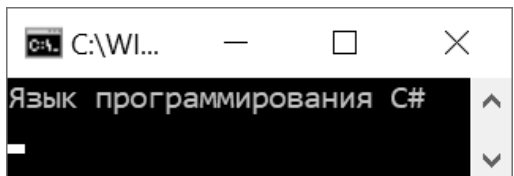
b - = a равносильно записи **b = b - a;**

Пример 3. Составной оператор присваивания

Выражение (запись в традиционной форме)	Выражение (запись с помощью составного оператора)
a = a+34	a +=34
a = a - 5	a-= 5
c = c / 2	c/=2
c = c % 2	c%=2
d =d*3	d*=3
d=d*a	d*=a

Организация ввода/вывода

Пример 4. Вывод сообщения на экран

Оператор	Результат
<code>Console.Write("Язык программирования C#");</code>	
<code>Console.WriteLine("Язык программирования C#");</code>	

Пример 5. Ввод с клавиатуры значения переменной

Ввод целого числа

Вариант №1:

```
string b;           // b - описание переменной для ввода значений
b =Console.ReadLine(); // введенное с клавиатуры значение записывается в
                      // переменную b
int F = Convert.ToInt32( b ); // 1) значение в b "конвертируется" (преобразовывается)
// из строкового типа в указанный тип; в данном случае в целый тип.
                      // 2) преобразованное значение присваивается переменной F.
```

Вариант №2:

```
Console.WriteLine("Введите число F" );  
int F = Convert.ToInt32(Console.ReadLine());
```

Вариант №3:

```
string b= Console.ReadLine();  
int F = int.Parse(b); // Строковое представление числа b преобразовывается  
// в эквивалентное ему 32-х битовое знаковое целое число F с помощью метода Parse,  
// который реализован для всех числовых типов данных. Если преобразование выполнить  
// невозможно, то генерируется исключение, работа программы прерывается.
```

Ввод вещественного числа

Вариант №1:

```
Console.WriteLine("Введите вещественное число:" );  
string b = Console.ReadLine();  
double A = Convert.ToDouble(b);
```

Вариант №2:

```
Console.WriteLine("Введите вещественное число:" );  
double A = Convert.ToDouble(Console.ReadLine());
```

Вариант №3:

```
Console.WriteLine("Введите вещественное число:" );
```

```
string b= Console.ReadLine();  
double A =double.Parse(b);
```

Ввод строки

```
Console.WriteLine("Введите строку");  
string s = Console.ReadLine();
```

Пример 6. Вывод значений на экран

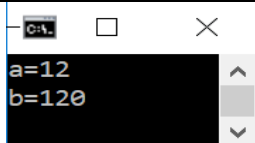
Исходные данные	Метод Console.Write	Результат
c= 2;	Console.Write("c=" + c);	c=2
c= 2;	Console.Write("c=" + (c+10));	c=12
a = 12; b = 13;	Console.Write("a=" + a); Console.Write(" b=" + b);	a = 12 b = 13
a = 12; b = 13;	Console.Write("a=" + a); Console.WriteLine(); Console.Write("b=" + b);	a = 12 b = 13
a = 12; b = 13;	Console.Write("a=" + a+"\nb=" + 12);	a = 12 b = 13
a = 12; b = 13;	Console.WriteLine("Вначале выводится число A={0}, затем B={1} и потом их сумма = {2}", a, b, a + b);	Вначале выводится A=12, затем B=13 и, потом их сумма

Исходные данные	Метод Console.Write	Результат
		=25
D=14.312; L=190	Console.Write("{0:G}", D); Console.Write("{0:G}", L);	14.312 190
A=12.6	Console.Write("A={0,10:N2}", A);	A= 12,60

Console.Write("A={0,10:N2}", A); // 10 символов на вывод числа (включая запятую), 2 из 10 - на вывод дробной части

Подробнее форматный вывод в [Приложении 3](#)

Пример 7. Использование управляющих символов

Оператор	Результат
<pre>int a = 12; Console.WriteLine("a=" + a + "\nb=" + 120); // \n - начало новой строки</pre>	

Список основных управляющих последовательностей в [Приложении 4](#)

Структура программы на языке программирования C#

using System;

//Директива using System - разрешает использовать имена стандартных

//классов из пространства имен System непосредственно (без указания имени пространства)

namespace ConsoleApplication1

//Ключевое слово namespace создает для проекта собственное пространство имен,

// названное по умолчанию ConsoleApplication1

{

class Program

//с ключевого слова class начинается Описание класса

//за которым следуют его имя, а в фигурных скобках — список элементов класса (его данных

// и функций, называемых также методами)

// В данной заготовке программы один класс, которому по умолчанию задано имя Program.

{

static void Main(string[] args)

//Каждое приложение должно содержать метод Main — с него начинается выполнение программы.

{

// текст программы

}

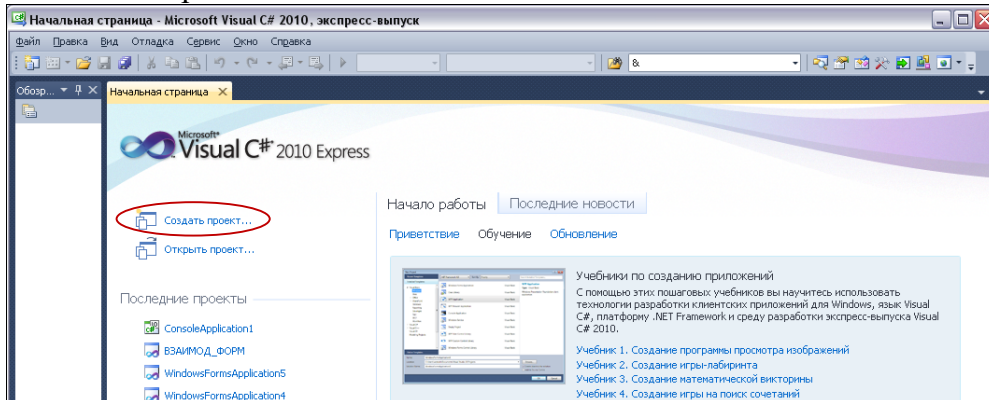
}

}

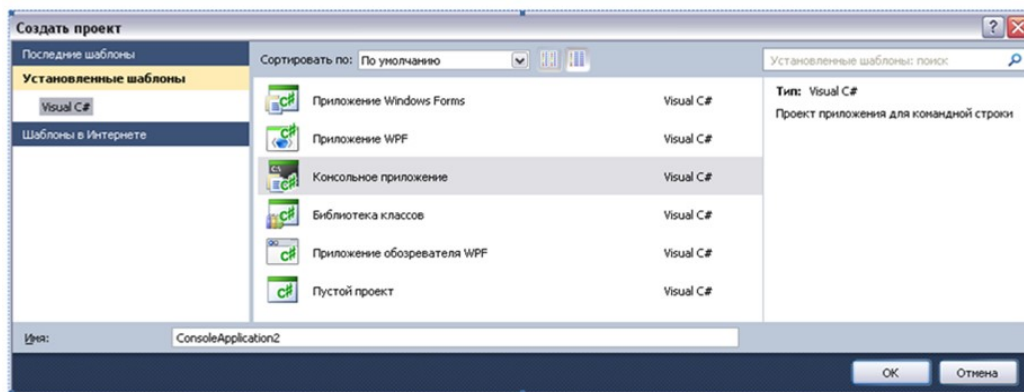
Задания для выполнения в аудитории

Задание №1. Работа со средой программирования Microsoft Visual 2010.

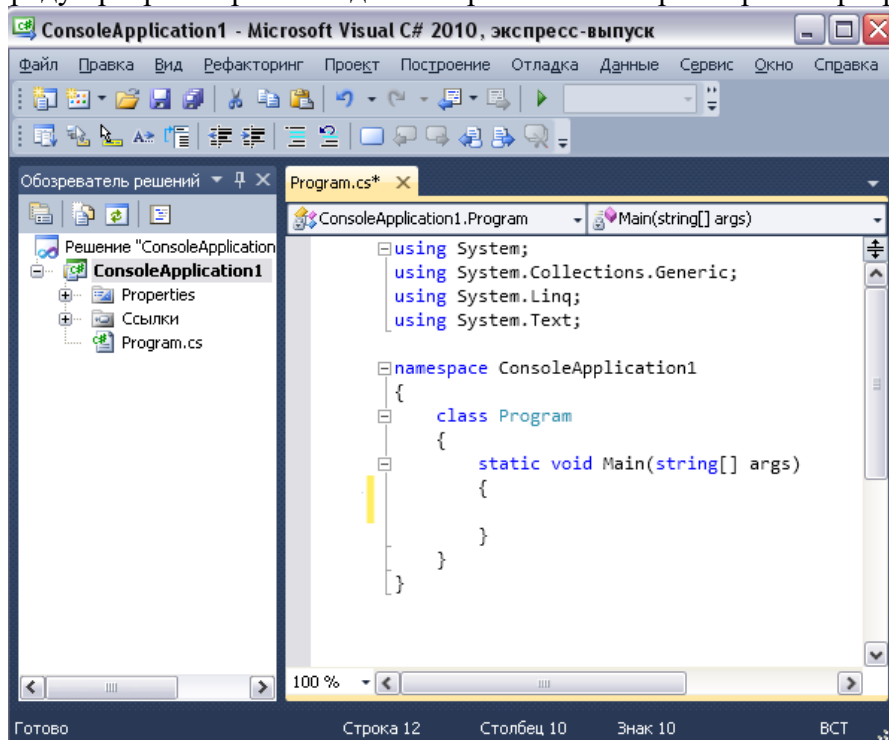
1. Запустите среду программирования Microsoft Visual 2010 Express.
2. Выберите "Создать проект":



3. Выберите "Консольное приложение" и введите имя проекта (по умолчанию имя проекта останется ConsoleApplication1):

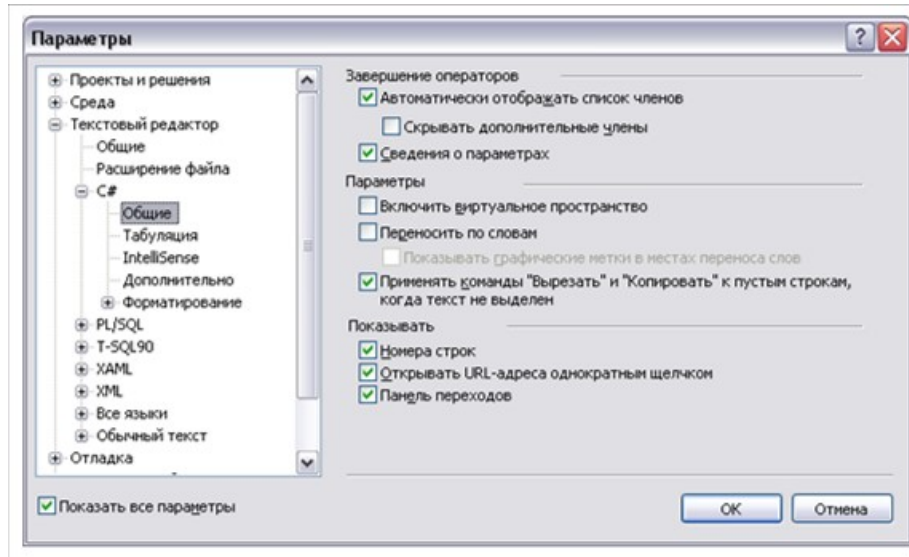


4. Настройте среду программирования для отображения номеров строк в программном коде.

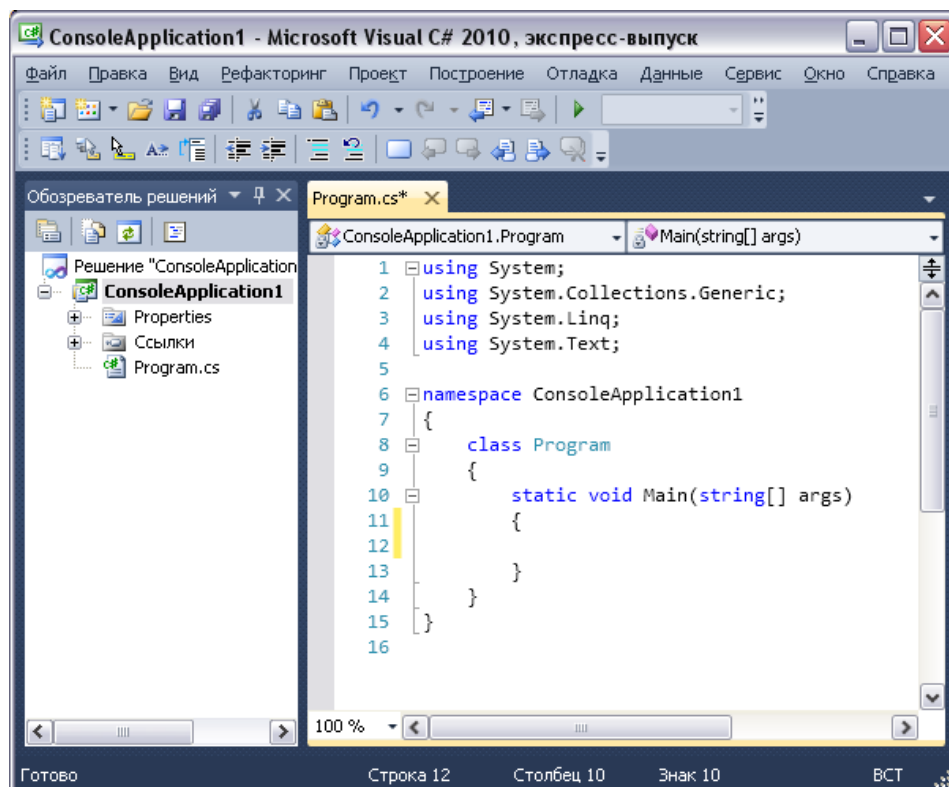


Для этого:

- выберите пункт меню **Сервис \ Параметры**
- в левой нижней части в появившегося диалогового окна отметьте знаком "галочка" пункт **"Показать все параметры"**
- перейдите: **Текстовый редактор \ C# \ Общие**
- в области "Показать" отметьте знаком "галочка" пункт **"Номера строк"** :



В окне редактора программного кода вводится текст программы:



4. Введите следующий текст программы в строку 12:

`Console.WriteLine("Задание №1. Первая программа.");`

5. Запустите программу на отладку: меню **Отладка \ Начать отладку** или нажать клавишу F5 (или щелкнуть по соответствующей пиктограмме)

6. Дополните программу следующим кодом:

```
// значение префиксного выражения:
int r;
r = 6;
Console.WriteLine(" значение префиксного выражения:");
Console.WriteLine(" ++r = " + (++r));

// значение постфиксного выражения:
int t = 6;
Console.WriteLine(" значение постфиксного выражения:");
Console.WriteLine("до ++ t = " + (t++) + " после ++ t=" + t);
```

7. Запустите программу на отладку и проанализируйте полученный результат.

Задание №2. Составьте программу для вычисления значения следующего выражения:

$$K = (138 * X + Y) / (X - 24) - (X * Y - 67) / (78 + X).$$

Требования к программе:

- X и Y целого типа;
- ввод значений переменных X и Y организуйте с клавиатуры;
- вывод результата представьте в следующем виде, например: X=0 Y=0 K = 0,8589;
- значение переменной K выведите с шестью знаками после запятой.

Подготовьте тестовые примеры для проверки правильности вычисления значения выражения.

Задание №3. Дано двузначное число. Составьте программу для выполнения следующих действий над данным числом:

1. нахождение числа десятков
2. нахождение числа единиц
3. нахождение суммы цифр числа
4. нахождение произведения цифр числа
5. получение числа, образованного при перестановке цифр исходного числа.

Требования к программе:

- ввод двузначного числа организуйте с клавиатуры
- вывод результата по каждому действию над числом, каждый результат с новой строки и соответствующим текстовым сопровождением, например: "Число десятков = ... ".

Задания для самостоятельной работы студента (СРС)

Задание №1. Разработайте программу для нахождения значений следующих выражений. Примеры использования математических функций (методы класса Math) смотрите в Приложении 5. Подготовьте тестовые примеры для проверки правильности вычисления значения выражений, с помощью табличного процессора.

Результат работы сохраните в файлах: *Задание1.cs* и *Тесты.xls*

Вариант	Выражение 1	Выражение 2
1	$Y = \cos^2\left(\frac{4}{7}\pi - a\right) + \cos^2\left(\frac{12}{7}\pi + \frac{a}{3}\right)$	$X = \frac{\cos 2a}{\sqrt{2\sin a}} + \operatorname{ctg} 2a$
2	$Y = \frac{(R+1)\sqrt{R} - (T+1)\sqrt{T}}{\sqrt{R^5} + RT - R^3 - R}$	$X = \frac{\sin 5b + 1}{\cos 5b + 1}$
3	$Y = \frac{\sin 2a + \sin 3a - \sin 4a}{\cos 2a + 1 + \cos 3a}$	$X = \frac{1}{\sqrt{2R+3}}$
4	$Y = \frac{\sqrt{R+3\sqrt{T^3-3}}}{\sqrt{T^2-3} + \sqrt[3]{R}}$	$X = \operatorname{tg}\left(\frac{3\pi}{2} - a\right) + 0.5$
5	$Y = \frac{\sin a + \cos(2b+a)}{\cos a - \cos(2b+a)} - 1$	$X = \sqrt{\frac{R-4}{R+4T}}$
6	$Y = \frac{\sqrt{(5T+2)^3 + 15T}}{5\sqrt{T} - \frac{3}{\sqrt{T}}}$	$X = \frac{1 - \operatorname{tg} a}{1 + \operatorname{tg} a}$
7	$Y = \frac{\cos 3a}{\cos a + 1} + \frac{\cos a}{1 - \sin 2a}$	$X = \frac{1}{\sqrt{2R}} + \frac{1}{\sqrt[3]{T}}$
8	$Y = \left(\frac{a+1}{\sqrt{3}} + \frac{a}{\sqrt{2a}} - \frac{1}{a^3}\right) - \frac{\sqrt[4]{a}}{a+1}$	$X = \frac{\sqrt{3}}{2} \cos\left(\frac{a}{2}\right)$
9	$Y = \sin^2\left(\frac{3\pi}{2}\right) + 3\cos^2 a - \frac{1}{4}\operatorname{tg} a + 1$	$X = (2\sqrt{2} \sin 2a) \operatorname{ctg}\left(\frac{\pi}{4} + 2a\right)$
10	$Y = \frac{\sqrt{3T-2\sqrt{T^2+2}}}{\sqrt{T^2+2} + \sqrt[3]{T}} - 4$	$X = (-5 \sin(\frac{R+T}{2})) \cos(a+b)$
11	$Y = \frac{\sin 3a + \sin 5a - \sin 7a}{\cos a + 1 - 2 \sin 4a}$	$X = \frac{\sqrt{R} - \sqrt{T}}{R}$
12	$Y = \frac{1+a^2+a^3}{3a+2} + 2 - \frac{\sqrt[3]{a^5}}{a+1}$	$X = -3 \cos^2\left(\frac{a-b}{3}\right) + \sin^2\left(\frac{1}{2}\pi - \frac{a}{2}\right)$
13	$Y = \left(\sin\left(\frac{\pi}{2} + 2a\right)\right) - \frac{1}{3} \cos(\pi + b)$	$X = \frac{4\sqrt{R} - 2\sqrt{T}}{R+T}$
14	$Y = \sin^4 a + \operatorname{tg}\left(\frac{2a}{3}\right) - \frac{1}{5} \cos^2 a + 2$	$X = \sqrt{\frac{R^3 + 3T}{T + \sqrt[3]{R}}}$

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
2. Встроенные типы (справочник по С#). URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/built-in-types> Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM .
3. Основы языка С# / ИНТУИТ Национальный открытый университет. URL: <https://www.intuit.ru/studies/courses/1139/250/lecture/6426?page=4>
4. Состав языка и типы данных/ ИНТУИТ Национальный открытый университет. URL: <http://www.intuit.ru/studies/courses/629/485/lecture/11001?page=2>.
5. Таблица встроенных типов (Справочник по С#). URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/built-in-types-table>

Приложение 1

Встроенные типы (справочник по С#)

Название типа	Тип	Биты	Диапазон значений
Логический	bool	8	истина/ложь
Целые типы	sbyte	8	от -128 до 127
	short	16	от -32768 до 32767
	int	32	от -2147483648 до 2147483647
	long	64	от -9223372036854775808 до 9223372036854775807
	byte	8	от 0 до 255
	ushort	16	от 0 до 65535
	uint	32	от 0 до 4294967295
	ulong	64	от 0 до 18446744073709551615
Символьный	char	16	символ
Вещественный	float	32	от -3.402823e38 до 3.402823e38
	double	64	от -1.79769313486232e308 до 1.79769313486232e308
Финансовый	decimal	128	от -79228162514264337593543950335 до 79228162514264337593543950335
Строковый	string	-	строка символов Unicode

Приложение 2

Некоторые операции языка C#

Операция	Описание
+	Унарный плюс
-	Арифметическое отрицание
*	Умножение
/	Деление
!	Логическое отрицание
++x	Префиксный инкремент
--x	Префиксный декремент
x++	Постфиксный инкремент
x--	Постфиксный декремент
=	Простое присваивание
*=	Умножение с присваиванием
/=	Деление с присваиванием
%=	Остаток от деления с присваиванием
+=	Сложение с присваиванием
-=	Вычитание с присваиванием
<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно
!=	Не равно
&	Поразрядное И
^	Поразрядное исключающее ИЛИ
	Поразрядное ИЛИ
&&	Логическое И
	Логическое ИЛИ
(тип) x	Преобразование типа

Приложение 3

Форматы вывода

Формат задается с помощью его строк. Спецификация формата следующая: {N, M: Axx}, где N указывает позицию элемента в списке выводимых переменных (нумерация начинается с нуля); M – задает ширину области, в которую будет помещено форматированное значение, если M отсутствует или отрицательно, значение будет выровнено влево, в противном случае — вправо; Axx — является необязательной строкой формирующих кодов, которые используются для управления форматированием чисел, даты и времени, денежных знаков и т.д.

Строка формата имеет следующий вид: Axx, где A — описатель формата, а xx — описатель точности. Описатель формата управляет типом форматирования, применяемым к числовому значению, а описатель точности – количеством значащих цифр или десятичных знаков форматированного результата.

Поддерживаемые строки основных стандартных форматов:

Формат	Описание формата	Исходные данные	Метод Console.Write	Результат
Общий формат	G или g	B=26.567 T=20	Console.Write("{0: G }", B); Console.Write("{0: G }", T);	26.567 20
Десятичный формат	D или d	B=26	Console.Write("{0: D5 }", B);	26000
Формат с фиксированной запятой	F или f	B=26.567 T=20	Console.Write("{0: F5 }", B); Console.Write("{0: F5 }", T);	26,56700 20,00000
—	—	B=26.567 T=20	Console.Write("{0: F0 }", B); Console.Write("{0: F0 }", T);	27 20
Числовой формат	N или n	B=26.567 T=20	Console.Write("{0: N }", B); Console.Write("{0: N }", T);	26,57 20,00
—	—	B=26.567 T=20	Console.Write("{0: N0 }", B); Console.Write("{0: N0 }", T);	27 20
—	—	B=26.567 T=20	Console.Write("{0: N5 }", B); Console.Write("{0: N5 }", T);	26,56700 20,00000
Инженерный формат	E или e	B=26.567 T=20	Console.Write("{0: E5 }", B); Console.Write("{0: E5 }", T);	2,65670E+001 2,00000E+001

Приложение 4

Для управления выводом на экран в С# используются *управляющие последовательности*.

Список основных управляющих последовательностей:

- \a Звуковой сигнал (звонок).
- \b Возврат на одну позицию.
- \f Подача страницы (для перехода к началу следующей страницы).
- \n Начало новой строки.
- \r Возврат каретки (Return).
- \t Горизонтальная табуляция.
- \v Вертикальная табуляция.
- \0 Нуль-символ.
- \' Одинарная кавычка (апостроф).
- \" Двойная кавычка.
- \\ Обратная косая черта.

Приложение 5

Некоторые методы класса Math

Имя метода	Описание	Тип результата	Пример Вызов метода	Результат работы метода
Cos	Косинус (Угол в радианах)	double	<code>double r = Math.Cos(3.14); Console.WriteLine("r="+r);</code>	r=-0,99999873172754
Sin	Синус (Угол в радианах)	double	<code>double r = Math.Sin(3.14); Console.WriteLine("r="+r);</code>	r=0,00159265291648683
Exp	Экспонента	double	<code>double r = Math.Exp(0.5); Console.WriteLine("r="+r);</code>	r=1,64872127070013
Max	Максимальное из двух чисел	перегружен	<code>double X = 4.586; double Y = 12.784; double Z = Math.Max(X, Y); Console.WriteLine("Z=" + Z);</code>	Возвращает большее из двух чисел двойной точности с плавающей запятой Z=12,784
Max	Максимальное из двух чисел	перегружен	<code>int X = 1234; int Y = 5678; int Z = Math.Max(X, Y); Console.WriteLine("Z=" + Z);</code>	Возвращает большее из двух 32-битовых целых чисел Z=5678
Min	Минимальное из двух чисел	перегружен	<code>int X = 1234; int Y = 5678; int Z = Math.Min(X, Y); Console.WriteLine("Z=" + Z);</code>	Возвращает меньшее из двух 32-битовых целых чисел Z=1234
Pow	Возведение в степень	double	<code>double F = Math.Pow(4, 2); Console.WriteLine("F=" + F);</code> Или: <code>Console.WriteLine("F=" + Math.Pow(4.2, 2.1));</code>	Возвращает указанное число, возведенное в указанную степень F=16 F=20,362
Sqrt	Квадратный корень	double	<code>double F=Math.Sqrt(16); Console.WriteLine("F=" + F);</code> Или: <code>Console.WriteLine("F=" + Math.Sqrt(4.3));</code>	Возвращает квадратный корень из указанного числа F=4 F=2,07364413533277
Round	Округление	перегружен	<code>double X=123.4567891011; double X1=Math.Round(X,3); Console.WriteLine("X1=" + X1);</code>	Округляет значение двойной точности с плавающей запятой до заданного количества дробных разрядов:

Имя метода	Описание	Тип результата	Пример Вызов метода	Результат работы метода
			Или: <code>double X = 123.4567891011;</code> <code>double X1 = Math.Round(X);</code> <code>Console.WriteLine("X1=" + X1);</code>	X1=123,457 Округляет заданное число с плавающей запятой двойной точности до ближайшего целого: X1=123
Log	Натуральный логарифм	double	<code>double r = Math.Log(2.7);</code> <code>Console.WriteLine("r="+r);</code>	r=0,993252
Log10	Десятичный логарифм	double	<code>double r = Math.Log10(10);</code> <code>Console.WriteLine("r="+r);</code>	r=1
Pi	Значение числа π	double	<code>double X = Math.PI;</code> <code>Console.WriteLine("Pi=" + X);</code> Или: <code>Console.WriteLine("Pi=" + Math.PI);</code>	Pi=3,14159265358979 Pi=3,14159265358979
E	Число e	double	<code>double X = Math.E;</code> <code>Console.WriteLine("E=" + X);</code> Или: <code>Console.WriteLine("E=" + Math.E);</code>	E=2,71828182845905 E=2,71828182845905

Приложение 6

Синтаксис описания класса:

```
[атрибуты][спецификаторы]class_имя_класса[:список_родителей]
{
тело_класса
}
```

[] – необязательные составляющие.

[атрибуты] – задают дополнительную информацию о классе. По умолчанию класс имеет атрибут доступа `internal`. Чтобы сделать класс доступным классам другого проекта, его нужно объявить с атрибутом `public`.

[спецификаторы] – определяют свойства класса, доступность класса для других элементов программы.

class – ключевое слово

имя_класса – задается программистом по общим правилам C#.

[:список_родителей] – базовые классы (предки)

тело_класса: – список описаний его элементов.

```
{
[статические поля - константы]; – неизменяемые значения, связанные с классом;
[поля]; – содержат данные класса;
[конструкторы]; – реализуют действия по инициализации класса или его экземпляров;
[ деструкторы]; – определяют действия, которые нужно выполнить до того, как экземпляр класса(объект) будет уничтожен;
[индексаторы]; – обеспечивают возможность доступа к элементам класса по их порядковому номеру;
[методы]; – реализуют вычисления или другие действия, выполняемые классом;
[события]; – элемент класса, позволяющий ему посылать другим; объектам уведомления об изменении своего состояния.
[классы (структуры, делегаты, интерфейсы, перечисления)]
}
```

Пример класса *Program*:

```
using System;
namespace ConsoleApplication1
{
    class Program
    { // тело класса
        static void Main(string[] args)
        {
            // тело метода
        }
    }
}
```

ОПЕРАТОРЫ ВЫБОРА

ОПЕРАТОР "if"

Условный оператор if позволяет осуществлять выбор одного варианта действий из двух возможных.

Синтаксис оператора if:

- полная форма условного оператора:
if (Выражение) { <Оператор 1> }[else { <Оператор 2>}]
- сокращенная форма условного оператора:
if (Выражение) <Оператор >

Пример 1. Полная форма оператора if

Фрагмент кода	Результат работы
<pre>int i, j; i = 10; j = 11; if (i < j) Console.WriteLine("i < j"); else Console.WriteLine("i >= j"); Console.WriteLine("Работа фрагмента кода закончена");</pre>	<p>i < j</p> <p>Работа фрагмента кода закончена</p>
<pre>int i, j; i = 100; j = 11; if (i < j) { Console.WriteLine("Работа оператора, если условие истинно"); Console.WriteLine("i < j"); } else { Console.WriteLine("Работа оператора, если условие ложно"); Console.WriteLine("i >= j"); } Console.WriteLine("Работа фрагмента кода закончена");</pre>	<p>Работа оператора, если условие ложно</p> <p>i >= j</p> <p>Работа фрагмента кода закончена</p>

Пример 2. Сокращенная форма оператора if

Фрагмент кода	Результат работы
<pre>int i, j; i = 10; j = 11; if(i < j) Console.WriteLine("i < j"); Console.WriteLine("Работа фрагмента кода закончена");</pre>	<p>i < j</p> <p>Работа фрагмента кода закончена</p>
<pre>int i, j; i = 100; j = 11; if(i < j) Console.WriteLine("i < j"); Console.WriteLine("Работа фрагмента кода закончена");</pre>	<p>Работа фрагмента кода закончена</p>

Логический оператор:

&	Поразрядное И
^	Поразрядное исключающее ИЛИ
	Поразрядное ИЛИ
&&	Логическое И
	Логическое ИЛИ
!	Логическое отрицание НЕ

Результатом выполнения *оператора отношения* или *логического оператора* является логическое значение типа bool. Значения типа bool могут сравниваться только на равенство или неравенство, поскольку истинные (true) и ложные (false) значения не упорядочиваются.

Логические операторы &, |, ^ и ! поддерживают основные логические операции И, ИЛИ, исключающее ИЛИ и НЕ в соответствии с приведенной ниже таблицей истинности:

P	Q	P&Q	P Q	P^Q	!P
false	false	false	false	false	true
true	false	false	true	true	false
false	true	false	true	true	true
true	true	true	true	false	false

Пример 3. Использование логических операторов.

Фрагмент кода	Результат работы
<pre>bool b1, b2; b1 = true; b2 = false; if (b1 & b2) Console.WriteLine("Нельзя выполнить"); if (!(b1 & b2)) Console.WriteLine("!(b1 & b2) - true"); if (b1 b2) Console.WriteLine("b1 b2 - true"); if (b1 ^ b2) Console.WriteLine("b1 ^ b2 - true");</pre>	<pre>!(b1 & b2) - true b1 b2 - true b1 ^ b2 - true</pre>

Пример 4. Вывести на экран значение, принадлежащее диапазону: [-10 ; +6]

Фрагмент кода	Результат работы
<pre>int X=60; if ((X <= 6) & (X >= -10)) Console.WriteLine(" X="+X+" принадлежит диапазону "); else Console.WriteLine(" X="+X+" НЕ принадлежит диапазону ");</pre>	<pre>X=60 НЕ принадлежит диапазону</pre>

ОПЕРАТОР "?"

Оператор ? называют тернарным, так как для его работы требуются три операнда.

Общая форма оператора ?:

(Выражение 1) ? (Выражение 2) : (Выражение 3);

Выражение1 должно относиться к типу bool, а Выражение2 и Выражение3 — к одному и тому же типу.

Значение выражения ? определяется следующим образом:

- в начале вычисляется *Выражение1*.
- Если *Выражение1* «истинно», то вычисляется *Выражение2*, а полученный результат определяет значение всего выражения ? в целом.
- Если же *Выражение1* «ложно», то вычисляется *Выражение3*, и его значение становится общим для всего выражения ?.

Пример 5. Демонстрация работы оператора «?».

Фрагмент кода	Результат работы
<pre>int a = 23; int b = -20; a = (b < 0 ? 3 * b : 2 * b); // равносильно записи a = b < 0 ? 3 * b : 2 * b; Console.WriteLine("a=" + a);</pre>	a=-60

Так как $b = -20$, то **Выражение 1:** $-20 < 0$ – истинно, следовательно, вычисляется **Выражение 2:** $a = 3 * b = 3 * (-20) = -60$. Следовательно, $a = -60$.

ОПЕРАТОР "switch case"

Инструкция *switch* позволяет делать выбор одного варианта из множества альтернатив. Работа *switch*: значение выражения последовательно сравнивается с константами из заданного списка. При обнаружении совпадения для одного из условий сравнения выполняется последовательность операторов, связанная с этим условием.

Общая форма записи оператора switch:

switch (Выражение)

```
{  
case Константное выражение 1: < [ Список операторов ] > break;  
case Константное выражение 2: < [ Список операторов ] > break;  
...  
case Константное выражение N: < [ Список операторов ] > break;  
  
[ default: < Список операторов > break;]  
}
```

Тип "Выражения" может быть: char, byte, short, int, string. Использование switch-инструкции представлено в следующих примерах.

Пример 6. По введенному числу на экран выводится название дня недели.

```
Console.WriteLine("ВВЕДИТЕ ВАРИАНТ ДЕЙСТВИЯ:");  
Console.WriteLine("1- ВЫВОД названия первого дня недели; 2 - ВЫВОД названия второго  
дня недели и т.д.");  
Console.WriteLine();  
int p = Convert.ToInt32( Console.ReadLine());  
switch (p)  
{
```

```

case 1: Console.WriteLine("ВЫБРАН ВАРИАНТ 1: понедельник"); break;
case 2: Console.WriteLine("ВЫБРАН ВАРИАНТ 2: вторник"); break;
case 3: Console.WriteLine("ВЫБРАН ВАРИАНТ 3 : среда"); break;
case 4: Console.WriteLine("ВЫБРАН ВАРИАНТ 4 : четверг"); break;
case 5: Console.WriteLine("ВЫБРАН ВАРИАНТ 5 : пятница"); break;
case 6: Console.WriteLine("ВЫБРАН ВАРИАНТ 6 : суббота"); break;
case 7: Console.WriteLine("ВЫБРАН ВАРИАНТ 7 : воскресенье"); break;
default: Console.WriteLine(" нет варианта выбора"); break;
}

```

Пример 7. По номеру дня недели выводится сообщение о том, есть ли занятия по информатике.

Используется оператор if совместно с оператором case.

```

Console.WriteLine("ВВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ :");
Console.WriteLine("1- понедельник; 2 - вторник и т.д.");
Console.WriteLine();
int p1 = Convert.ToInt32(Console.ReadLine());

int p=0;
if ((p1 <= 2) | (p1 == 5)) p = 1;
if ((p1==3)|(p1==4)|(p1==6)) p = 2;

switch (p)
{
    case 1: Console.WriteLine(" Есть занятия по информатике."); break;
    case 2: Console.WriteLine(" Нет занятий по информатике."); break;
    default: Console.WriteLine(" нет варианта выбора"); break;
}
Console.WriteLine();
Console.WriteLine(" Нажмите 'Enter' для выхода из программы");

```

Пример 8. По номеру дня недели выводится сообщения о том, есть ли занятия по информатике.

Используется сокращенная форма записи оператора case.

```

Console.WriteLine("ВВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ :");
Console.WriteLine("1- понедельник; 2 - вторник и т.д.");
Console.WriteLine();
int p = Convert.ToInt32(Console.ReadLine());

switch (p)
{
    case 1:
    case 2:
    case 4: Console.WriteLine(" Есть занятия по информатике."); break;
    case 3:
    case 5: Console.WriteLine(" Нет занятий по информатике."); break;
    default: Console.WriteLine(" нет варианта выбора"); break;
}
Console.WriteLine();
Console.WriteLine(" Нажмите 'Enter' для выхода из программы");

```

или:

```
Console.WriteLine("ВВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ :");
Console.WriteLine("1- понедельник; 2 - вторник и т.д.");
Console.WriteLine();
int p = Convert.ToInt32(Console.ReadLine());

switch (p)
{
    case 1:case 2:case 4: Console.WriteLine(" Есть занятия по информатике."); break;
    case 3:case 5: Console.WriteLine(" Нет занятий по информатике."); break;
    default: Console.WriteLine(" нет варианта выбора"); break;
}
Console.WriteLine();
Console.WriteLine(" Нажмите 'Enter' для выхода из программы");
```

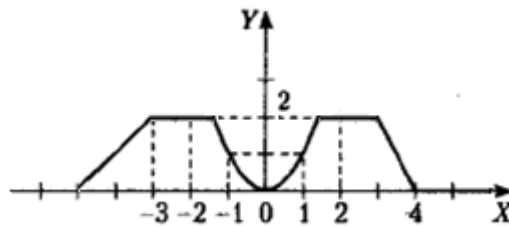
Задания для выполнения в аудитории

Задание №1. Выполните следующие задания с использованием оператора If. Для каждого из заданий подготовьте необходимые тесты.

1.1. Для данного вещественного x найти значение следующей функции $f(x)$:

$$f(x) = \begin{cases} -x, & \text{если } x \leq 0, \\ x^2, & \text{если } 0 < x < 2, \\ 4, & \text{если } x \geq 2 \end{cases}$$

1.2. Напишите программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика.



Задание №2. Выполните следующее задание с использованием оператора Case.

Дан номер месяца — целое число в диапазоне 1–12 (1 — январь, 2 — февраль и т. д.).

Составить программу для вывода на экран название соответствующего времени года («зима», «весна», «лето», «осень»).

Указание к выполнению: используйте сокращенную форму записи «Case».

Задания для самостоятельной работы студента (СРС)

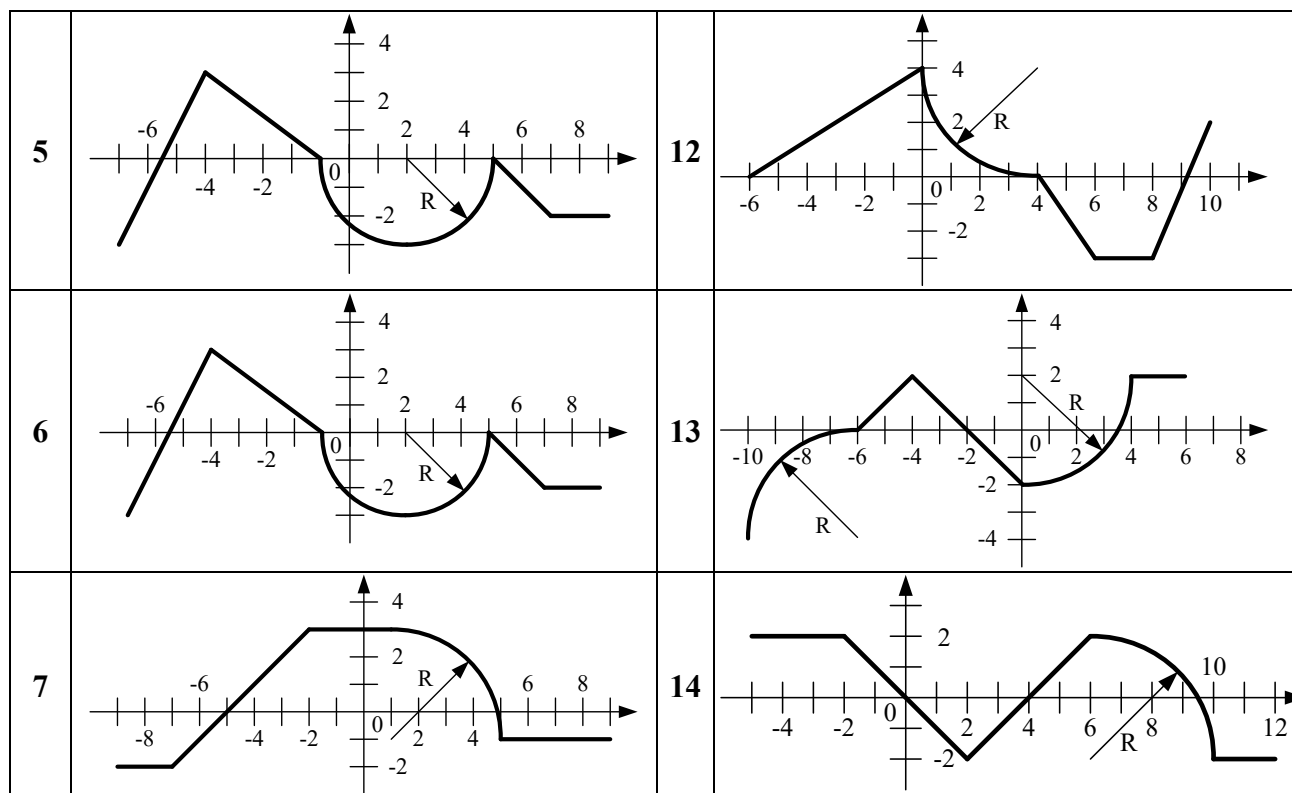
Задание №1. Выполните следующие задания с использованием оператора If. Подготовьте необходимые тесты.

1.1. Для данного вещественного x найти значение следующей функции f , принимающей значения целого типа:

$$f(x) = \begin{cases} 0, & \text{если } x < 0, \\ 1, & \text{если } x \text{ принадлежит } [0, 1), [2, 3), \dots, \\ -1, & \text{если } x \text{ принадлежит } [1, 2), [3, 4), \dots \end{cases}$$

1.2. Напишите программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика.

Ва- ри- ант	График	Ва- ри- ант	График
1		8	
2		9	
3		10	
4		11	



Задание №2. Выполните следующее задание с использованием оператора Case:

Дано целое число в диапазоне 20–69, определяющее возраст (в годах). Вывести строку-описание указанного возраста, обеспечив правильное согласование числа со словом «год», например: 20 — «двадцать лет», 32 — «тридцать два года», 41 — «сорок один год».

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
2. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM

ОПЕРАТОРЫ ЦИКЛА

Операторы цикла используются для организации многократно повторяющихся вычислений.

К операторам цикла относятся:

- цикл с пред-условием *while –do*;
- цикл с пост-условием *do – while*;
- цикл с параметром *for*;
- цикл перебора *foreach*.

ЦИКЛЫ: "while" И "do - while "

Цикл while (цикл с пред-условием)

Общая форма записи цикла *while*:

while (Выражение)

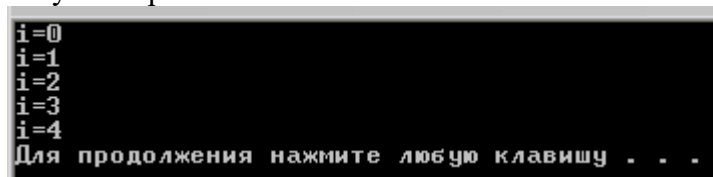
```
{  
// Операторы тела цикла;  
}
```

Выражение может быть как простым, так и сложным логическим выражением. **Телом цикла** может быть простой или составной оператор.

Пример 1. Фрагмент кода для вывода на экран значений параметра цикла. В качестве «Выражения» используется простое логическое выражение.

```
int i = 0; // начальные значения параметра цикла;  
while (i < 5)  
{  
    Console.WriteLine("i="+i);  
    i = i + 1; // изменения параметра цикла;  
}
```

Результат работы:



```
i=0  
i=1  
i=2  
i=3  
i=4  
Для продолжения нажмите любую клавишу . . .
```

Пример 2. Фрагмент кода для решения задачи нахождения значения факториала числа. В качестве «Выражения» используется «нестрогое» логическое неравенство.

```
int j = 4; // значение числа, факториал которого нужно найти;  
int Fk = 1, n = 1; // начальные значения: факториала (Fk) и параметра цикла (n);
```

while (n <= j)

```
{  
    Fk *= n; // Fk = Fk * n;  
    n++;  
}  
// вывод Fk.
```

Пример 3. Фрагмент кода для решения задачи нахождения значения факториала числа. В качестве "Выражения" используется "строгое" логическое неравенство.

```
int j = 4; // значение числа, факториал которого нужно найти
int Fk = 1, n = 1; // начальные значения: факториала (Fk) и параметра цикла (n)

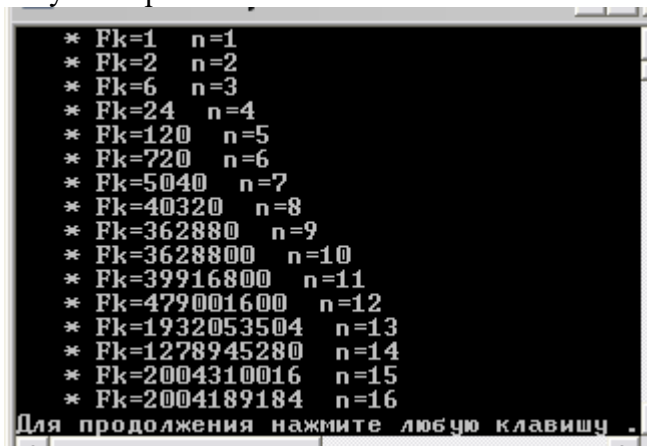
while (n < j)
{
    n++;
    Fk *= n; // Fk = Fk * n
}
// вывод Fk
```

Пример 4. Выводятся на экран все значения факториала, входящие в диапазон типа int.

Наибольшее возможное значение типа System.Int32 определяется константой MaxValue. Фрагмент кода:

```
int y = int.MaxValue;
int Fk = 1, n = 1; // начальные значения: факториала (Fk) и параметра цикла (n);
while ((Fk <= y) & (Fk > 0))
{
    Console.WriteLine(" * Fk=" + Fk + " n=" + n);
    n++;
    Fk *= n; // Fk = Fk * n;
}
```

Результат работы:



```
* Fk=1    n=1
* Fk=2    n=2
* Fk=6     n=3
* Fk=24   n=4
* Fk=120  n=5
* Fk=720  n=6
* Fk=5040 n=7
* Fk=40320 n=8
* Fk=362880 n=9
* Fk=3628800 n=10
* Fk=39916800 n=11
* Fk=479001600 n=12
* Fk=1932053504 n=13
* Fk=1278945280 n=14
* Fk=2004310016 n=15
* Fk=2004189184 n=16
Для продолжения нажмите любую клавишу .
```

Цикл do ... while (цикл с пост-условием)

Общая форма записи цикла *do...while*:

```
do
{
    // Операторы тела цикла;
}while (Выражение)
```

Выражение может быть как простым, так и сложным логическим выражением. **Телом цикла** может быть простой или составной оператор.

Пример 5. Фрагмент кода для решения задачи нахождения значения факториала числа.

```
int j=4, Fk = 1, n = 1; // j=4-значение числа, факториал которого нужно найти;  
do  
{  
    Fk *= n;  
    n++;  
} while (n <= j); // при n=5-выход из цикла
```

Пример 6. Фрагмент кода демонстрирует увеличение на 1 значения переменной В до тех пор, пока не будет введено значение параметра цикла, равное 5.

```
int i,B=0;  
do  
{  
    B = B+1;  
    i= Convert.ToInt32( Console.ReadLine());  
} while (i != 5);
```

Задания для выполнения в аудитории

Задание №1.

Составьте программу для построения таблицы значений функции: $H=2y^2+4y-0,3$ для $y=0,1; 0,11; 0,12; \dots; 0,2$.

(Используйте циклы **do while** и **while**).

Оформите результат в виде таблицы, например:

Y	H
0.1	0.120
...	...

Для организации вывода используйте следующий вывод, например:

```
Console.WriteLine("  Y   |   H   ");
Console.WriteLine("-----");
Console.WriteLine("  + Y+  |  + H +  ");
```

Задание №2. Составьте программу для вычисления значение выражения $1!+2!+3!+\dots+N!$ с использованием циклов: **while** и **do while**. Значение переменной N введите с клавиатуры.

Задания для самостоятельной работы студента (СРС)

Задание №1. Выполните задание с использованием цикла **while**.

Разработайте консольное приложение для вычисления и вывода на экран в виде таблицы значения функции, заданной графически, в диапазоне от $X_{\text{нач}}$ до $X_{\text{кон}}$ с шагом dx .

Требования к программе:

- $X_{\text{нач}}$, $X_{\text{кон}}$ – задайте вводом с клавиатуры;
- шаг dx задайте в программе таким образом, чтобы была возможность проверить все ветви программы;
- значения Y выведите на экран с тремя знаками после запятой;
- таблицу обеспечьте: заголовком, например «Таблица значений функции» и шапкой.

Ва- ри- ант	График	Ва- ри- ант	График
1		8	
2		9	
3		10	
4		11	
5		12	

Ва- ри- ант	График	Ва- ри- ант	График
6		13	
7		14	

Задание №2. Выполните задание с использованием цикла **do - while**.

Вариант	Задание
1	Разработайте консольное приложение для вывода на экран наибольшего из всех отрицательных чисел вводимой последовательности N целых чисел.
2	Разработайте консольное приложение для вывода на экран все целые числа из диапазона от A до B ($A \leq B$), оканчивающиеся на цифру X или Y.
3	Разработайте консольное приложение для вывода на экран всех целых чисел из диапазона от A до B ($A \leq B$), оканчивающихся на любую четную цифру.
4	Разработайте консольное приложение для вывода на экран только положительных целых чисел из диапазона от A до B ($A \leq B$).
5	Разработайте консольное приложение для вывода на экран таблицы перевода 5, 10, 15, ..., 120 долларов в рубли по текущему курсу (значение курса введите с клавиатуры)..
6	Разработайте консольное приложение для вывода на экран всех целых чисел из диапазона от A до B, кратные трем ($A \leq B$).
7	Разработайте консольное приложение для вывода на экран таблицы стоимости для 10, 20, 30, ..., 100 штук товара, при условии, что одна штука товара стоит X рублей (значение X введите с клавиатуры).
8	Разработайте консольное приложение для вывода на экран всех четных чисел из диапазона от A до B, кратные трем ($A \leq B$).
9	Разработайте консольное приложение для вывода на экран кубы всех целых чисел из диапазона от A до B ($A \leq B$), в обратном порядке.
10	Разработайте консольное приложение для вывода на экран только отрицательные четные числа из диапазона от A до B ($A \leq B$).
11	Разработайте консольное приложение для вывода на экран все трехзначные числа, которые начинаются и заканчиваются на одну и ту же цифру.
12	Разработайте консольное приложение для вывода на экран все двухзначные числа, в записи которых все цифры разные.
13	Разработайте консольное приложение для вывода на экран таблицы перевода расстояний в дюймах в сантиметры для значений 2, 4,6, ... ,12 дюймов (1 дюйм=25,4 мм).
14	Разработайте консольное приложение для вывода на экран все трехзначные числа, в которых хотя бы две цифры повторяются.

ЦИКЛ "for"

Цикл for (цикл с параметром)

Общая форма записи цикла *for*:

```
for (<Инициализация>; <Выражение>; <Модификация>)
{
// Операторы тела цикла;
}
```

Инициализация используется для объявления величин, которые применяются в цикле, и для присвоения им начального значения (счетчик, параметр цикла).

Выражение определяет условие выполнения цикла: если результатом выражения является истина, цикл выполняется. Тип **Выражение** - bool.

Модификация используется для изменения параметра цикла.

Телом цикла может быть простой или составной оператор.

Пример 1. Цикл for с одним параметром.

Фрагмент кода	Результат работы фрагмента кода
<pre>int I; for (I = 0; I < 5; I = I + 1) Console.WriteLine("REZULT: " + I);</pre>	<pre>REZULT: 0 REZULT: 1 REZULT: 2 REZULT: 3 REZULT: 4</pre>

Пример 2. Цикл for с двумя параметрами: i и j.

Фрагмент кода	Результат работы фрагмента кода
<pre>int i, j; for(i=0, j=10; (i <= j)&(i<4); i++, j--) // &-и; - или Console.WriteLine("i=" + i + " j=" + j);</pre>	<pre>i=0 j=10 i=1 j=9 i=2 j=8 i=3 j=7</pre>

Пример 3. Работа цикл for с использованием условного выражения.

Фрагмент кода	Результат работы фрагмента кода
<pre>int i, j; bool d = false; for(i=0, j=100; !d; i++, j--) { if (i*i >= j) d = true; Console.WriteLine("i=" + i + "\t i*i= " + i*i + "\t j=" + j); }</pre>	<pre>i=0 i*i= 0 j=100 i=1 i*i= 1 j=99 i=2 i*i= 4 j=98 i=3 i*i= 9 j=97 i=4 i*i= 16 j=96 i=5 i*i= 25 j=95 i=6 i*i= 36 j=94 i=7 i*i= 49 j=93 i=8 i*i= 64 j=92 i=9 i*i= 81 j=91 i=10 i*i= 100 j=90</pre>

При **i=10** и **j=90**, значение **i*i= 100**, что, в свою очередь, больше чем j (j=90).

Следовательно, **d = true** и цикл закончит работу.

Пример 4. В описании цикла отсутствует часть цикла "Модификация".

Фрагмент кода	Результат работы фрагмента кода
<pre>int i; for(i = 0; i <= 5;) { Console.WriteLine("i=" + i); i++; // инкрементировать переменную управления циклом }</pre>	<pre>i=0 i=1 i=2 i=3 i=4 i=5</pre>

Пример 5. Расчет суммы чисел от 1 до 5. В записи цикла отсутствует тело цикла.

Фрагмент кода	Результат работы фрагмента кода
<pre>int Y; int sum = 0; for(Y = 1; Y <= 5; sum += Y++); Console.WriteLine("Sum = " + sum);</pre>	<pre>Sum = 15</pre>

Пример 6. Расчет суммы чисел от 1 до 5. Объявление управляющих переменных в цикле for.

Фрагмент кода	Результат работы фрагмента кода
<pre>int sum = 0; for(int i = 1; i <= 5; i++) { sum += i; } // Переменная i доступна только в цикле [for(int i = 1...)] // Вне цикла переменная i недоступна. Console.WriteLine("Sum = " + sum);</pre>	<pre>Sum = 15</pre>

Задания для выполнения в аудитории

Задание №1. Составьте программу для построения таблицы значений функции:

$$H=2y^2+4y-0,3$$

для $y = 0,1; 0,11; 0,12; \dots; 0,2$.

Используйте цикл **for**. Оформите результат в виде таблицы, например:

Y	H
0.1	0.120
...	...

Для организации вывода используйте следующий вывод, например:

```
Console.WriteLine("  Y  |  H  ");
Console.WriteLine("-----");
Console.WriteLine("  + Y + |  + H +  ");
```

Задание №2. Составьте программу для вычисления факториала числа, введенного с клавиатуры.

Используйте конструкцию цикла **for** «без тела цикла» с:

- объявлением управляющих переменных обычным образом;
- объявлением управляющих переменных в цикле **for**.

Задание №3. Организуйте в цикле ввод пяти чисел целого типа. Предусмотрите вывод на экран только тех чисел, которые кратны 3.

Используйте:

- конструкцию цикла **FOR** «с отсутствием части цикла»;
- изменение переменной, управляющей циклом, – в теле цикла.

Задания для самостоятельной работы студента (СРС)

Задание №1. Составьте программу, которая выводит сообщение о результате попадания точки, с заданными координатами, в заштрихованную область.

Требования к программе:

- используйте конструкцию цикла for;
- количество "выстрелов по мишени", введите с клавиатуры;
- сообщение о результате попадания точки, выведите в виде текстового сообщения.

Ва- ри- ант	Область	Ва- ри- ант	Область
1		8	
2		9	
3		10	

Ва- ри- ант	Область	Ва- ри- ант	Область
4		11	
5		12	
6		13	
7		14	

ВЛОЖЕННЫЕ ЦИКЛЫ

Если тело цикла содержит структуру, которая является циклом, то такой цикл называется кратным (вложенным). В этом случае различают внешний и внутренний циклы. Кратность циклов не ограничена.

Вложенными могут быть циклы любых типов: *while*, *do while*, *for*.

Примеры использования вложенных циклов:

При- мер	Фрагмент кода	Результат работы фрагмента кода	При- мер	Фрагмент кода	Результат работы фрагмента кода
Количество повторений внутреннего цикла не зависят от номера прохода внешнего цикла					
1	<pre>for (int i = 1; i <= 5; ++i) { for (int j=1; j<=5; ++j) { Console.Write(" 5 "); } Console.WriteLine(); }</pre>	<pre>5 5</pre>	3	<pre>for (int i = 1; i <= 5; ++i) { for (int j=1; j<=5; ++j) { Console.Write(j); } Console.WriteLine(); }</pre>	<pre>12345 12345 12345 12345 12345</pre>
2	<pre>for (int i = 1; i <= 5; ++i) { for (int j=1; j<=5; ++j) { Console.Write(i); } Console.WriteLine(); }</pre>	<pre>11111 22222 33333 44444 55555</pre>	4	<pre>for (int i = 1; i <= 5; ++i) { for (int j=1; j<=5; ++j) { Console.Write(j); } Console.WriteLine(); } Console.WriteLine(i); //Console.WriteLine(j);</pre>	<p>Ошибка. "i" не существует в текущем контексте</p>
Количество повторений внутреннего цикла зависит от номера прохода внешнего цикла					
1	<pre>for (int i = 1; i <= 5; ++i) { for (int j = 1; j <= i; ++j) { Console.Write(j); } Console.WriteLine(); }</pre>	<pre>1 12 123 1234 12345</pre>	2	<pre>for (int i = 1; i <= 5; ++i) { for (int j = i; j <= 5; ++j) { Console.Write(j); } Console.WriteLine(); }</pre>	<pre>12345 2345 345 45 5</pre>

При- мер	Фрагмент кода	Результат работы фрагмента кода	При- мер	Фрагмент кода	Результат работы фрагмента кода
3	<pre> for (int i = 1; i <= 5; ++i) { for (int j = 1; j <= i; ++j) { Console.Write(i); } Console.WriteLine(); } </pre>	<pre> 1 22 333 4444 55555 </pre>	4	<pre> for (int i = 1; i <= 5; ++i) { for (int j = i; j <= 5; ++j) { Console.Write(i); } Console.WriteLine(); } </pre>	<pre> 11111 2222 333 44 5 </pre>

Задания для выполнения в аудитории

Задание №1. Составьте программу для вычисления значения выражения $1!+2!+3!+\dots+n!$ с использованием вложенных циклов FOR.

Задание №2. Составьте программу для вывода на экран чисел в следующем виде:

Вариант №	Задание	Вариант №	Задание
1	6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	2	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5

Количество строк и столбцов введите с клавиатуры.

Задания для самостоятельной работы студента (СРС)

Задание №1. Разработайте программу для вывода на экран таблицы умножения в следующем виде, например для таблицы 4x4.

Таблица умножения 4 X 4

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

Требования к программе:

- размер таблицы задается вводом с клавиатуры;
- для вывода таблицы используйте форматный вывод.

Задание №2. Разработайте программу для вывода на экран следующих изображений (с использованием вложенных циклов и форматного вывода).

Вариант	Задание	Вариант	Задание
1	1 * 2 * 2 * 3 * 3 * 3 * 4 * 4 * 4 * 4 * 5 * 5 * 5 * 5 * 5 *	8	8,2 8,2 8,2 8,2 8,2 8,4 8,4 8,4 8,4 8,4 8,6 8,6 8,6 8,6 8,6 8,8 8,8 8,8 8,8 8,8 9,0 9,0 9,0 9,0 9,0
2	1 1 2 1 2 3 1 2 3 4 1 2 3 4 5	9	1 2 3 4 5 1 2 3 4 1 2 3 1 2 1

Ва- ри- ант	Задание	Ва- ри- ант	Задание
3	<pre> -4 -3 -2 -1 0 -3 -2 -1 0 -2 -1 0 -1 0 0 </pre>	10	<pre> 4 4 3 3 4 3 2 2 3 2 4 2 1 1 2 1 3 1 4 1 </pre> <p>Используйте вывод параметров внутреннего и внешнего циклов</p>
4	<pre> * 1 * 1 * 2 * 1 * 2 * 3 * 1 * 2 * 3 * 4 * 1 * 2 * 3 * 4 * 5 </pre>	11	<pre> 7 6 6 5 5 5 4 4 4 4 3 3 3 3 3 2 2 2 2 2 2 1 1 1 1 1 1 1 </pre>
5	<pre> 4,1 4,1 4,1 4,1 4,1 4,3 4,3 4,3 4,3 4,3 4,5 4,5 4,5 4,5 4,5 4,7 4,7 4,7 4,7 4,7 4,9 4,9 4,9 4,9 4,9 </pre>	12	<pre> 6 7 8 9 10 11 5 6 7 8 9 4 5 6 7 3 4 5 2 3 1 </pre>
6	<pre> 0 1 2 3 4 0 1 2 3 0 1 2 0 1 0 </pre>	13	<pre> 6 6 6 6 6 6 5 5 5 5 5 4 4 4 4 3 3 3 2 2 1 </pre>
7	<pre> 7,0 7,1 7,2 7,3 7,4 7,0 7,1 7,2 7,3 7,4 7,0 7,1 7,2 7,3 7,4 7,0 7,1 7,2 7,3 7,4 7,0 7,1 7,2 7,3 7,4 </pre>	14	<pre> & & & & & & & & & & & & & & & </pre>

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Кудрина Е.В., Огнева М.В., Портенко М.С. Программирование на языке C#: разработка консольных приложений. М.: Национальный открытый университет "ИНТУИТ", 2016.
2. Павловская Т.А. C#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
3. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM

РЕКУРРЕНТНЫЕ СООТНОШЕНИЯ

Формулы, устанавливающие связь предыдущего и последующего элементов последовательности, называются - *рекуррентными соотношениями*. Каждый следующий член последовательности может выражаться через несколько предыдущих членов последовательности.

Последовательность задана рекуррентно, если для нее определено следующее рекуррентное соотношение и заданы первые l членов:

$$a_k = f(a_{k-1}, a_{k-2}, \dots, a_{k-l}), k > l$$

, где a_1, a_2, \dots, a_n – числовая последовательность.

ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ С ПОМОЩЬЮ РЕКУРРЕНТНЫХ СООТНОШЕНИЙ

При решении задач на вычисление сумм (произведений) членов конечных или бесконечных последовательностей, вычисления пределов последовательностей, – наибольшую трудность представляет получение рекуррентной формулы. Далеко не во всех задачах закономерность, связывающая предыдущий и последующий члены последовательности, очевидна, как, например, в последовательности: $X, X^2, X^3, X^4, \dots, X^n$

Примеры рекуррентной последовательности:

- арифметическая прогрессия. Рекуррентное соотношение: $a_n = a_{n-1} + d$, где d – разность прогрессии;
- геометрическая прогрессия. Рекуррентное соотношение: $b_n = b_{n-1} * q$, где q – знаменатель прогрессии;
- последовательности Фибоначчи. Рекуррентное соотношение: $a_n = a_{n-1} + a_{n-2}$, где $a_1 = a_2 = 1$;

Пример 1. Общий член ряда известен.

Дан следующий ряд: $1; \frac{1}{2!}; \frac{1}{3!}; \dots \frac{1}{n!}$.

Получите *рекуррентное соотношение* для данного ряда и разработайте программу для вычисления *суммы* n заданных элементов ряда.

Решение:

1. Получение рекуррентного соотношения.

1.1. Пронумеровать элементы ряда и определить значение первого элемента:

$$1; \frac{1}{2!}; \frac{1}{3!}; \dots \frac{1}{n!}$$

$$i = 1 \quad 2 \quad 3 \quad n$$

$$a_1 = 1$$

1.2. Записать формулу n -го члена ряда: $a_n = \frac{1}{n!}$

1.3. Записать формулу $(n+1)$ -го члена ряда: $a_{n+1} = \frac{1}{(n+1)!}$

1.4. Разделить (n) -ый член на $(n-1)$ -й член ряда:

$$\frac{a_n}{a_{n-1}} = \frac{(n-1)!}{(n)!} = \frac{1}{n}$$

В результате получаем множитель (знаменатель последовательности), который связывает следующий член последовательности с предыдущим.

Рекуррентное соотношение:

$$a_n = a_{n-1} \cdot (1/n)$$

$$\text{Проверка: } a_1=1; \quad a_2= a_1 \cdot (1/(2))= 1 \cdot (1/(2))=1/1 \cdot 2=1/2!$$

2. Вычисление n - го элемента последовательности, используя рекуррентную формулу

$$a_n = a_{n-1} \cdot (1/n)$$

Фрагмент кода:

```
double a=1, n=3; //начальные значения последовательности
for (int i=1; i<=n; i++)
{
a=a*1/(i); // вычисление следующего элемента последовательности
}
```

Пример 2. Общий член ряда неизвестен.

$$\frac{x^1}{2} ; \frac{x^2}{4} ; \frac{x^3}{8} ; \frac{x^4}{16} \dots\dots\dots$$

Дан следующий ряд:

Получите рекуррентное соотношение для данного ряда и разработайте программу для вычисления n - го элемента последовательности. .

Решение:

1. Получение рекуррентного соотношения.

Вариант 1

1. 1. Пронумеровать элементы ряда и определить значение первого элемента:

$$\frac{x^1}{2} ; \frac{x^2}{4} ; \frac{x^3}{8} ; \frac{x^4}{16} \dots\dots\dots$$

$$i = 1 \quad 2 \quad 3 \quad 4$$

$$a_1 = \frac{x^1}{2}$$

1.2. Найти:

$$\frac{a_2}{a_1} = \frac{x^2 \cdot 2}{4 \cdot x} = \frac{x}{2} ; a_2 = a_1 \cdot \frac{x}{2}$$

$$\frac{a_3}{a_2} = \frac{x^3 \cdot 4}{8 \cdot x^2} = \frac{x}{2} ; a_3 = a_2 \cdot \frac{x}{2}$$

$$\frac{a_4}{a_3} = \frac{X^4 8}{16 \cdot X^3} = \frac{X}{2} ; a_4 = a_3 \cdot \frac{X}{2}$$

В результате получим множитель (знаменатель последовательности), который связывает следующий член последовательности с предыдущим. Рекуррентное соотношение(формула):

$$a_n = a_{n-1} \cdot \frac{x}{2}$$

Вариант 2

$$\frac{x^n}{2^n}$$

1. Подобрать общий элемент ряда: $a_n = \frac{x^n}{2^n}$
2. Следовать пунктам 1.3. и 1.4. Примера 1.

2. Вычисление n - го элемента последовательности.

Фрагмент кода:

```
int x = 1, n=3;
double a=x / 2; //начальные значения последовательности и суммы
for (int i=1; i<=n; i++)
{
    a=a*x/2; // вычисление следующего элемента последовательности
}
```

Пример 3. Числитель и знаменатель элемента ряда рассматриваются отдельно.

Дан следующий ряд:

$$S_n = \frac{\sin(x)}{1} + \frac{\sin(x) + \sin(2x)}{2} + \frac{\sin(x) + \sin(2x) + \sin(3x)}{3} + \dots + \frac{\sin(x) + \dots + \sin(nx)}{n},$$

где n – натуральное число, x – вещественное число.

Получите рекуррентное соотношение для данного ряда и разработайте программу для вычисления суммы n заданных элементов ряда.

Решение:

1. Пронумеровать элементы ряда:

$$i = \frac{\sin(x)}{1} + \frac{\sin(x) + \sin(2x)}{2} + \frac{\sin(x) + \sin(2x) + \sin(3x)}{3} + \dots + \frac{\sin(x) + \dots + \sin(nx)}{n}$$

2. Знаменатель: значение знаменателя совпадает с номером элемента ряда.

3. Числитель: $ch_1 = \sin(x)$;
 $ch_2 = \sin(x) + \sin(2x)$
 $ch_2 == ch_1 + \sin(2x);$

$$ch_3 = \sin(x) + \sin(2x) + \sin(3x);$$

$$ch_3 = ch_2 + \sin(3x),$$

$$\text{где } ch_1 = ch_0 + \sin(1x) ; ch_0 = 0.$$

Используя рекуррентное соотношение получаем для числителя:

$$ch_0 = 0; ch_n = ch_{n-1} + \sin(nx), n - \text{номер элемента ряда}$$

4. Для суммы элементов ряда:

$$S_n = \frac{ch_1}{1} + \frac{ch_2}{2} + \frac{ch_3}{3} + \dots + \frac{ch_n}{n}$$

Расчет суммы с рекуррентным соотношением:

$$S_0 = 0; S_n = S_{n-1} + \frac{ch_n}{n}.$$

Фрагмент кода:

```
double ch = 0, s = 0; int n = 3, x = 1; //начальные значения последовательности и
суммы
for (int i = 1; i <= n; i++)
{
    ch += Math.Sin(i * x); // вычисление следующего элемента последовательности
    s += ch / i;
    Console.WriteLine("s={0:f3}", s);
}
```

Задания для выполнения в аудитории

Задание №1. Реализуйте **Пример №3** (раздел Организация вычислений с помощью рекуррентных соотношений), в виде консольного приложения.

Требования к программе:

- количество элементов ряда и значение X введите с клавиатуры;

- результат работы приложения выведите на экран в следующем виде, например, для N=3, X=1:

N	X	S
1	1	0.841
2	1	1.717
3	1	2.347

- значения суммы выведите с тремя знаками после запятой (используйте форматный вывод).

Задание №2. Составьте программу для вычисления значения функции F(x) на заданном отрезке, с указанным шагом, с использованием рекуррентных соотношений.

Вариант	F(x)
1	$F(x) = 1 - \frac{x}{2 \cdot 7} + \frac{x^2}{4 \cdot 14} - \frac{x^3}{8 \cdot 21} + \frac{x^4}{16 \cdot 28} - \dots, x \in [0; 0.9]$
2	$F(x) = 1 - \frac{x^2}{1 \cdot 3 \cdot 4} + \frac{x^4}{2 \cdot 4 \cdot 5} - \frac{x^6}{3 \cdot 5 \cdot 6} + \frac{x^8}{4 \cdot 6 \cdot 7} - \dots, x \in [0.2; 0.7]$
3	$F(x) = 1 + \frac{x^3}{3 \cdot 2} + \frac{x^5}{5 \cdot 2^2} + \frac{x^7}{7 \cdot 2^3} + \dots, x \in [0; 0.99]$
4	$F(x) = 1 + \frac{x}{1 \cdot 4} - \frac{x^2}{2 \cdot 5} + \frac{x^3}{3 \cdot 6} - \frac{x^4}{4 \cdot 7} + \dots, x \in [0.1; 0.9]$
5	$F(x) = 1 - \frac{x^3}{3 \cdot 4^2} + \frac{x^5}{4 \cdot 5^2} - \frac{x^7}{5 \cdot 6^2} + \dots, x \in [0; 0.8]$
6	$F(x) = 1 + \frac{x^2}{2 \cdot 4} + \frac{x^3}{4 \cdot 6} + \frac{x^4}{6 \cdot 8} + \dots, x \in [0.2; 0.6]$
7	$F(x) = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \frac{1}{7x^7} - \dots, x \in [-3; -2]$
8	$F(x) = 1 - \frac{x}{2 \cdot 7} + \frac{x^2}{4 \cdot 14} - \frac{x^3}{8 \cdot 21} + \frac{x^4}{16 \cdot 28} - \dots, x \in [0; 0.9]$
9	$F(x) = 1 - \frac{x^2}{1 \cdot 3 \cdot 4} + \frac{x^4}{2 \cdot 4 \cdot 5} - \frac{x^6}{3 \cdot 5 \cdot 6} + \frac{x^8}{4 \cdot 6 \cdot 7} - \dots, x \in [0.2; 0.7]$
10	$F(x) = 1 + \frac{x^3}{3 \cdot 2} + \frac{x^5}{5 \cdot 2^2} + \frac{x^7}{7 \cdot 2^3} + \dots, x \in [0; 0.99]$

Вариант	F(x)
11	$F(x) = 1 + \frac{x}{1 \cdot 4} - \frac{x^2}{2 \cdot 5} + \frac{x^3}{3 \cdot 6} - \frac{x^4}{4 \cdot 7} + \dots, x \in [0.1; 0.9]$
12	$F(x) = 1 - \frac{x^3}{3 \cdot 4^2} + \frac{x^5}{4 \cdot 5^2} - \frac{x^7}{5 \cdot 6^2} + \dots, x \in [0; 0.8]$
13	$F(x) = 1 + \frac{x^2}{2 \cdot 4} + \frac{x^3}{4 \cdot 6} + \frac{x^4}{6 \cdot 8} + \dots, x \in [0.2; 0.6]$
14	$F(x) = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \frac{1}{7x^7} - \dots, x \in [-3; -2]$

Требования к программе:

- количество элементов ряда и значение X введите с клавиатуры;
- шаг $h = 0.1$ задается в программе;
- результат работы приложения выведите на экран в следующем виде:

N	X	F(x)
1	1	0.841
2	1	1.717
3	1	2.347

- значения суммы выведите с тремя знаками после запятой.

Указания к выполнению задания:

- обратите внимание: n - й элемент ряда неизвестен;
- используйте рекуррентное соотношение;
- для определения рекуррентного соотношения рассмотрите отдельно числитель и знаменатель:

1. Обратите внимание на нумерацию элементов ряда

$$F(x) = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \frac{1}{7x^7} - \dots, x \in [-3; -2].$$

1 2 3 4

2. Используйте связь номера элемента ряда со степенью переменной X.

3. Используйте связь номера элемента ряда с числами в знаменателе.

4. Запишите формулу для n - го элемента ряда.

5. Получите *рекуррентное соотношение* для данного ряда, используя

$$\frac{a_n}{a_{n-1}}$$

знаменатель последовательности: a_{n-1} .

Задания для самостоятельной работы студента (СРС)

Задание №1. Разработайте программу для вычисления $F(x)$:

$$F(x) = 1 + \frac{x}{1 \cdot 3} + \frac{x^2}{1 \cdot 3 \cdot 5} + \frac{x^3}{1 \cdot 3 \cdot 5 \cdot 7} + \dots, x \in [0.05; 0.95]$$

Количество элементов ряда и шаг изменения X , определить самостоятельно.

Задание №2

1. Рассмотрите материал по книге Павловской Т.А., с. 83-86.
2. Выполните следующее задание:

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от $X_{\text{нач}}$ до $X_{\text{кон}}$ с шагом dx с точностью ϵ .

Требования к программе:

- Таблицу снабдить заголовком и шапкой.
- Каждая строка таблицы должна содержать:
 - значение аргумента, значение функции и количество просуммированных членов ряда.
- Предусмотрите вывод сообщения о невозможности вычислить ряд с заданной точностью.

Вариант	Задание
1	$\ln x = \sum_{n=0}^{\infty} \frac{(-1)^n (x-1)^{n+1}}{(n+1)} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \dots, 0 < x < 2.$
2	$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots, x < \infty.$
3	$\operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \dots, x > 1.$
4	$\ln x = 2 \sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}} = 2 \left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \dots \right), x > 0.$
5	$\ln x = \sum_{n=0}^{\infty} \frac{(x-1)^{n+1}}{(n+1)x^{n+1}} = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \dots, x > \frac{1}{2}.$
6	$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots, x < \infty.$
7	$e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots, x < \infty.$

Вариант	Задание
8	$\ln(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n} = -\left(x + \frac{x^2}{2} + \frac{x^3}{3} + \dots\right), -1 \leq x < 1.$
9	$\ln(x+1) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} - \dots, -1 < x < 1.$
10	$\ln \frac{1+x}{1-x} = 2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots\right), x < 1.$
11	$e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots, x < \infty.$
12	$\operatorname{arctg} x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots, x < 1.$
13	$\frac{\sin x}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots, x < \infty.$
14	$\ln \frac{x+1}{x-1} = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots\right), x > 1.$
15	$\operatorname{arcctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} x^{2n+1}}{2n+1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} - \dots, x < 1.$

3. Оформите отчет о выполнении работы.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Кудрина Е.В., Огнева М.В., Портенко М.С. Программирование на языке С#: разработка консольных приложений. М.: Национальный открытый университет "ИНТУИТ", 2016.
2. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
3. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM .

МЕТОДЫ

Метод представляет собой блок кода, содержащий набор инструкций.

В C# все инструкции выполняются в контексте метода. Метод Main является точкой входа для каждого приложения C#, и вызывается он средой CLR при запуске программы.

В C# любой метод является методом какого-либо класса. Например, метод **Main**(точка входа в программу) класса **Program** :

```
class Program
{
    static void Main(string[] args)
    {
    }
}
```

Место метода в структуре класса:

тело класса - список описаний его элементов

```
{
    [Статические поля - константы]; - неизменяемые значения, связанные с классом;
    [Поля]; - содержат данные класса;
    [конструкторы]; - реализуют действия по инициализации класса или его экземпляров;
    [деструкторы]; - определяют действия, которые нужно выполнить до того, как
                    экземпляр класса(объект) будет уничтожен;
    [индексаторы]; - обеспечивают возможность доступа к элементам класса по их
                    порядковому номеру;
    [методы]; - реализуют вычисления или другие действия, выполняемые классом;
    [события]; - элемент класса, позволяющий ему посылать другим объектам
                уведомления об изменении своего состояния;
    [классы (структуры, делегаты, интерфейсы, перечисления)].
}
```

Синтаксис описания метода:

```
[атрибуты][спецификаторы] тип имя_метода([параметры])
{
    тело метода
}
```

[] – *необязательные составляющие.*

[атрибуты]	– задают дополнительную информацию о методе;
[спецификаторы]	– определяют доступность метода для других элементов программы; чаще всего для методов задается спецификатор public, ведь методы составляют интерфейс класса, поэтому они должны быть доступны;
тип	– определяет, значение какого типа вычисляется с помощью метода; если метод не возвращает никакого значения, то в его заголовке задается тип void, а оператор return отсутствует;
имя_метода	– задается программистом по общим правилам C#;
[параметры]	– используются для обмена информацией с методом; параметр

представляет собой локальную переменную, которая при вызове метода принимает значение соответствующего аргумента; область действия параметра – весь метод; параметры, описываемые в заголовке метода, определяют множество значений аргументов, которые можно передавать в метод;

{тело метода} – задает действия, выполняемые методом; чаще всего представляет собой блок – последовательность операторов.

Например, метод **Main** класса **Program**:

<pre>class Program { static void Main(string[] args) // заголовок метода { //тело метода } }</pre>	<p>заголовок метода: спецификатор - static(метод доступен на уровне класса); тип - void; имя_метода - Main; параметры - string[] ar</p>
--	--

СТАТИЧЕСКИЕ МЕТОДЫ

Статическим называется метод, который существует в классе и обеспечивают функциональность классов и программы в целом. В заголовке статических методов указывается модификатор доступа static (метод доступен на уровне класса).

I. Метод, не возвращающий значения

Пример 1. Метод без параметров. Вычисление периметра прямоугольника.

Решение:

```
class Program
{
    static void SL1() //описание метода SL без параметров
    {
        int a = 1; int b = 2;
        int p = 2 * (a + b);
        Console.WriteLine("p={0}", p);
    }
    static void Main(string[] args)
    {
        SL1();// вызов метода SL1
    }
}
```

Метод не возвращает в основной метод Main никакого результата, поэтому тип возвращаемого значения void.

В теле метода отсутствует оператор return. В основном методе Main, метод **SL1** вызывается один раз.

При необходимости методы могут быть вызваны столько раз, сколько нужно для решения задачи.

Пример 2. Метод с параметрами (с входными параметрами - параметрами-значениями).

Вычисление периметра прямоугольника.

Решение:

```
1. class Program
2. {
3.     static void SL1(int a, int b) //описание метода SL.
4.     {
5.         int p = 2 * (a + b);
6.         Console.WriteLine("p={0}", p);
7.     }
8.     static void Main(string[] args)
9.     {
10.        SL1(1,2); // вызов метода SL1
11.    }
12. }
```

Метод не возвращает в основной метод Main никакого результата, поэтому тип возвращаемого значения void.

Строка №3 – заголовок метода; **int a, int b** - формальные параметры метода. Строка №10 – вызов метода SL1(1, 2), числа 1 и 2 – фактические параметры метода.

II. Метод, возвращающий значения

Пример 3. Метод без входных параметров.

Вычисление периметра прямоугольника.

Решение:

```
1. class Program
2. {
3.     static int SL1()
4.     {
5.         int a = 1; int b = 2;
6.         int p = 2 * (a + b);
7.         return p;
8.     }
9.     static void Main(string[] args)
10.    {
11.        int rez= SL1();
12.        Console.WriteLine("p={0}", rez);
13.    }
14. }
```

Для возвращения результата работы в вызывающий метод, необходимо:

- в заголовке метода указать тип результата работы метода. В данном примере строка №3 **int SL1()** - результата работы метода SL1 целого типа;
- в теле метода указать оператор передачи управления – return, с указанием имени той переменной, значение которой будет передано в вызывающий метод. В данном примере в строка №7 значение переменной p будет передано в основной метод Main;
- в строке вызова метода, указать переменную, которой присваивается результат работы вызываемого метода. В данном примере в строке №11: результат работы метода SL1() присваивается целочисленной переменной rez.

Оператор *return* завершает выполнение метода и передает управление в точку вызова метода.

Пример 4. Метод с входными параметрами.
Вычисление периметра прямоугольника.

Решение:

```
1. class Program
2. {
3.     static int SL1(int a, int b)
4.     {
5.         int p = 2 * (a + b);
6.         return p;
7.     }
8.     static void Main(string[] args)
9.     {
10.        int a = 1, b = 2;
11.        int rez= SL1(a,b);
12.        Console.WriteLine("p={0}", rez);
13.    }
14. }
```

Вызов метода SL1(a,b) располагается в строке №11. Результат работы метода SL1 присваивается целочисленной переменной rez.

Пример 5. Метод с входными параметрами. Вызов метода в операторе вывода.
Вычисление периметра прямоугольника.

Решение:

```
1. class Program
2. {
3.     static int SL1(int a, int b)
4.     {
5.         int p = 2 * (a + b);
6.         return p;
7.     }
8.     static void Main(string[] args)
9.     {
10.        int a = 1, b = 2;
11.        Console.WriteLine("p={0}", SL1(a, b));
12.    }
13. }
```

Вызов метода SL1(a,b) располагается в строке №11 (вывод результата).

III. Передача параметров по ссылке

Передача параметров по ссылке ref:

При передаче параметров по ссылке все изменения параметров в методе отражаются на фактических параметрах (переменной аргумента, используемой в вызове метода).

Пример 6. Метод `METHOD_1` реализует обмен значений двух переменных `z` и `w`. Передача параметров `z` и `w` в вызываемый метод осуществляется по ссылке *ref*.

```
1. class Program
2. {
3.     static void METHOD_1 (ref int z, ref int w)
4.     {
5.         int temp = z;
6.         z = w;
7.         w = temp;
8.     }
9.     static void Main(string[] args)
10.    {
11.        int z = 20, w = 50; // инициализация z и w.
12.        Console.WriteLine("результат до вызова метода METHOD_1: Z=" + z + " ; w=" + w);
13.
14.        METHOD_1(ref z, ref w); // ВЫЗОВ МЕТОДА METHOD_1.
15.        Console.WriteLine("результат после вызова метода METHOD_1: Z=" + z + " ; w=" + w);
16.    }
17. }
```

При использовании ссылки *ref* необходимо учитывать следующее:

- необходима инициализация параметров, передаваемых в метод (строка 11 фрагмента кода);
- использование ключевого слова *ref* при вызове метода (строка 14), и при описании метода (строка 3);
- измененные значения параметров в методе (строки 5,6,7) будут переданы в основной метод `Main`, в точку вызова (строка 14).

Передача параметров по ссылке out:

использовании

Пример 7. Метод `METHOD_2` реализует инициализацию переменных различных типов. Передача параметров в вызываемый метод осуществляется по ссылке *out*.

```
1. class Program
2. {
3.     static void METHOD_2(out int I, out string s1, out double D)
4.     {
5.         s1 = " РАБОТА МЕТОДА с параметром out;";
6.         I = 44;
7.         D = 0.555;
8.     }
9.     static void Main(string[] args)
10.    {
11.        int PRIMER; string str1; double D; // Описание переменных
12.        METHOD_2(out PRIMER, out str1, out D); // Вызов метода METHOD_2
13.
14.        Console.WriteLine(" str1=" + str1 + " PRIMER=" + PRIMER + " D=" + D);
15.    }
16. }
```

При использовании ссылки *out*, необходимо учитывать следующее:

-инициализация параметров, передаваемых в метод не требуется - достаточно описания (строка 11 фрагмента кода);

-использование ключевого слова *out* при вызове метода (строка 12) и при описании метода (строка 3);

-измененные значения параметров в методе (строки 5,6,7) будут переданы в основной метод Main, в точку вызова (строка 12).

Задания для выполнения в аудитории

Задание №1. Опишите класс дробей – чисел, являющихся отношением двух целых чисел.

Предусмотрите следующие методы:

1. *Сложения дробей* (входные параметры – четыре целых числа; метод, без передачи параметров).
2. *Вычитания дробей* (входные параметры – четыре целых числа; метод, с передачей параметров – результата вычитания дробей).
3. *Умножения деления дробей*, где входные параметры: числа для двух дробей; выходные параметры: исходные числа для дробей, результат умножения дробей, результат деления дробей.

Передача параметров осуществляется «по ссылке».

Разработайте программу, демонстрирующую все разработанные элементы класса.

Задания для самостоятельной работы студента (СРС)

Задание №1. Опишите класс, в котором будут находиться методы:

1. Метод, вычисляющий факториал числа, и возвращающий это значение.
2. Метод, определяющий сумму ряда: $0!+1!+2!+\dots+N!$. Сумма должна вычисляться до максимального значения типа `int` (используйте `int.MaxValue` – константа, представляющая максимальное значение типа).

Необходимые поля определите самостоятельно.

Составьте программу, демонстрирующую разработанные элементы класса.

Задание №2. Разработайте программу для расчета значения X, определив необходимый метод (методы) с входными и выходными параметрами.

Ва-ри-ант	Задание	Ва-ри-ант	Задание
1	$X = \frac{\sqrt{6} + 6}{2} + \frac{\sqrt{12} + 12}{2} + \frac{\sqrt{22} + 22}{2}$	8	$X = \frac{\sqrt{10} - 5}{\sqrt{5} + 10} + \frac{\sqrt{14} - 7}{\sqrt{7} + 14} + \frac{\sqrt{16} - 8}{\sqrt{8} + 16}$
2	$X = \frac{4 - \sqrt{4}}{0.3} + \frac{7 - \sqrt{7}}{0.3} + \frac{55 - \sqrt{55}}{0.3}$	9	$X = \frac{\sqrt{18} + 9}{\sqrt{9} + 18} + \frac{\sqrt{12} + 6}{\sqrt{6} + 12} + \frac{\sqrt{33} + 11}{\sqrt{11} + 11}$
3	$X = \frac{2 + \cos 2}{\cos 4 + 4} + \frac{1 + \cos 1}{\cos 4 + 4} + \frac{12 + \cos 12}{\cos 4 + 4}$	10	$X = \frac{15 - \sqrt{15}}{1.5} - \frac{3 - \sqrt{3}}{1.5} - \frac{12 - \sqrt{12}}{1.5}$
4	$X = \frac{\sqrt{10} - 5}{\sqrt{5} + 10} + \frac{\sqrt{14} - 7}{\sqrt{7} + 14} + \frac{\sqrt{16} - 8}{\sqrt{8} + 16}$	11	$X = \frac{4 - \sqrt{4}}{0.3} + \frac{7 - \sqrt{7}}{0.3} + \frac{55 - \sqrt{55}}{0.3}$
5	$X = \frac{1 - \sin 1}{4} + \frac{8 - \sin 8}{4} + \frac{11 - \sin 11}{4}$	12	$X = \frac{8 - \cos 8}{\cos 2 + 2} + \frac{32 - \cos 32}{\cos 2 + 2} + \frac{96 - \cos 96}{\cos 2 + 2}$
6	$X = \frac{\sqrt{18} + 9}{\sqrt{9} + 18} + \frac{\sqrt{12} + 6}{\sqrt{6} + 12} + \frac{\sqrt{33} + 11}{\sqrt{11} + 11}$	13	$X = \frac{1 - \sin 1}{\sin 4} - \frac{8 - \sin 8}{\sin 4} - \frac{11 - \sin 11}{\sin 4}$
7	$X = \frac{5 - \cos 6}{\cos 6 + 5} + \frac{91 - \cos 91}{\cos 4 + 4} + \frac{62 - \cos 62}{\cos 55 + 55}$	14	$X = \frac{25 + \sqrt{25}}{12.1} + \frac{19 + \sqrt{19}}{12.1} + \frac{48 + \sqrt{48}}{12.1}$

РЕКУРСИВНЫЕ МЕТОДЫ

Рекурсия — определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя.[4]

Практическое применение она нашла в работе информационных систем, физических экспериментах.

Рекурсивными ситуациями, или рекурсией в программировании, называют моменты, когда процедура или функция программы вызывает саму себя.

Рекурсивные версии многих методов выполняются медленнее, чем их итеративные варианты. Связано это с дополнительными затратами системных ресурсов, связанных с многократными вызовами методов. Большое количество рекурсивных обращений к методу может вызвать переполнение стека.

Рекурсивные методы

Рекурсивным называется метод, который вызывает в качестве вспомогательного метода сам себя.

Пример 1. Рекурсивный метод, вычисляющий факториал числа. (*Прямая рекурсия.*)
Вычисления проводятся на рекурсивном *возврате* (подъеме).

Решение:

Вычислить факториал возможно:

1. $0! = 1! = 1$, $n! = 1 * 2 * 3 \dots * n$.
2. $n! = (n-1)! * n$, для $n > 1$, при этом для $n = 0$ и $n = 1$ значение факториала равно 1.

```
static long F(long n) //рекурсивный метод
{
    if (n == 0 || n == 1) //нерекурсивная ветвь
    {
        return rez;
    }
    else
    {
        long rez = n * F(n - 1); //шаг рекурсии -
                                //вызов метода с другим параметром
        return rez;
    }
}
```

Вызов метода **F** из основного метода Main:

```
long f = F(5); //вызов метода F для вычисления 5!
```

Пример 2. Рекурсивный метод, вычисляющий факториал числа. (*Прямая рекурсия.*)
Вычисления проводятся на рекурсивном *спуске*.

Решение:

```
static int FS(int ym, int i, int m)
{
    ym=ym*i; // накопление факториала располагается до рекурсивного вызова
```

```

    if (i == m) return ym; //нерекурсивная ветвь
    else return FS(ym, i + 1, m); // шаг рекурсии – вызов метода с другим параметром
}

```

Вызов метода **FS** из основного метода **Main** может быть представлен следующим образом, например:

```
Console.WriteLine("FS="+FS(1,1,5)); // 5 - число, факториал которого рассчитывается
```

Пример 3. Рекурсивный метод, вычисляющий X^n .

Используется следующая рекурсивная формула:

$$x^n = \begin{cases} 1, & n=0 \\ 1/x^{|n|}, & n<0 \\ x \cdot x^{n-1}, & n>0 \end{cases}$$

```

static long STEPEN(long X, long n) //рекурсивный метод
{
    long rez;
    if (n == 0) //нерекурсивная ветвь
    {
        rez = 1;
        return rez;
    }
    else // рекурсивная ветвь
    {
        if (n > 0)
        {
            rez = X*STEPEN(X, (n - 1));
            return rez;
        }
        else //для отрицательной степени
        {
            rez = 1/STEPEN(X, Math.Abs(n));
            return rez;
        }
    }
}

```

Вызов метода из основного метода **Main**:

```
long st = STEPEN(3,5); //возведение числа 3 в 5-ю степень
```

Пример 4. Рекурсивный метод, не возвращающий значения.

Для заданного значения **n** вывести на экран **n** строк, в каждой из которых содержится **n** символов. Например, для **n=5** на экран нужно вывести следующее изображение:

```

#
# #
# # #
# # # #
# # # # #

```

Решение:

Рекурсивный метод:

```
static void Simvol_Rec(int n, int i)
{
    //если номер текущей строки не больше номера последней строки, то
    if (i <= n)
    {
        //выводим i символов в текущей строке
        for ( int j = 1; j <= i; j++)
        {
            Console.Write("# ");
        }
        Console.WriteLine();
        //рекурсивный переход к формированию строки с количеством символов на один больше
        Simvol_Rec(n, i + 1);
    }
}
```

Вызов метода из основного метода Main:

```
int n=5; //n – количество выводимых строк
int i=1; //i – количество символов в текущей строке (в первой строке один символ)
F_Rec(n, i);
```

Задания для выполнения в аудитории

Задание №1. Разработайте консольное приложение для реализации рекурсивного метода из примера 1.

Требования к программе:

- Предусмотрите вывод сообщения о **каждом рекурсивном вызове**, например
« Рекурсивный вызов **F** для $n = 3$ »
« Рекурсивный вызов **F** для $n = 2$ »

... ..

и т.д.

- Предусмотрите вывод сообщения о результате вычисления факториала числа, например:
« Вычисление факториала для $n = 1$ »
« Вычисление факториала для $n = 2$ »
.....

И т.д.

Задание №2.

2.1. Разработайте консольное приложение для реализации рекурсивного метода из примера №4.

Требования к программе:

1. Добавьте в программу отдельный метод для вывода на экран указанных символов.
2. Организуйте ввод с клавиатуры количества выводимых строк.

2.2. Внесите изменения в программу, которые позволят вывести на экран следующие изображения:

для $n=4$:

```
1
22
333
4444
```

для $n=5$; обратите внимание на количество символов в первой строке:

```
###
####
#####
```


Задания для самостоятельной работы студента (СРС)

Задание №1. Выполните задания, согласно Вашему варианту.

Вариант	Задание
<i>Разработайте рекурсивный метод, возвращающий значение:</i>	
1	для нахождения n-го члена последовательности $b_1 = -10, b_2 = 2, b_{n+2} = b_n - 6 \cdot b_{n+1}$
2	для нахождения количества цифр заданного натурального числа N.
3	для нахождения n-го члена последовательности $b_1 = 5, b_{n+1} = b_n / (n^2 + n + 1)$
4	для вычисления функции Аккермана для неотрицательных чисел n и m. Функция Аккермана определяется следующим образом: $A(n, m) = \begin{cases} m + 1, & \text{если } n = 0; \\ A(n - 1, 1), & \text{если } n \neq 0, m = 0; \\ A(n - 1, A(n, m - 1)), & \text{если } n > 0, m > 0. \end{cases}$
5	для нахождения числа сочетаний $C(n, m)$, используя следующие свойства: $C_n^0 = C_n^n = 1; C_n^m = C_{n-1}^m + C_{n-1}^{m-1} \text{ при } 0 < m < n$
6	для нахождения наибольшего общего делителя, рекурсивная формула $\text{НОД}(a, b) = \begin{cases} a, & \text{если } a = b; \\ \text{НОД}(a - b, b), & \text{если } a > b; \\ \text{НОД}(a, b - a), & \text{если } b > a. \end{cases}$
7	для нахождения числа a, для которого выполняется неравенство $2^{a-1} \leq n \leq 2^a$, где n – натуральное число. Число a определяется по следующей формуле: $a(n) = \begin{cases} 1, & n = 1; \\ a(n/2) + 1, & n > 1 \end{cases}$
8	для вычисления x^n (x–вещественное, $x \neq 0$, a n–целое) по формуле $x^n = \begin{cases} 1 & \text{при } n = 0, \\ 1/x^{ n } & \text{при } n < 0, \\ x \cdot x^{n-1} & \text{при } n > 0. \end{cases}$
<i>Разработайте рекурсивный метод, не возвращающий значение:</i>	
9	дано натуральное четное число n. Разработать рекурсивный метод для вывода на экран следующей картинки: <pre> * * * * * * * * * * * * * * * ... * </pre> <p style="text-align: center;">(n звездочек, через пробел)</p> <p>Количество звездочек (n) в первой строке введите с клавиатуры.</p>

Вари- ант	Задание
10	<p>дано натуральное число n. Разработать рекурсивный метод для вывода на экран следующей последовательности чисел:</p> <pre> 1 2 2 3 3 3 ... n n ... n </pre> <p>n - введите с клавиатуры. Между числами три пробела.</p>
11	<p>дано натуральное четное число n. Разработать рекурсивный метод для вывода на экран следующей картинки:</p> <pre> ***** (0 пробелов; N символов) ***** (1 пробел; N-1 символ) ***** (2 пробела; N-2 символа) ... * (N-1 пробел; 1 символ) </pre>
12	<p>дано натуральное четное число n. Разработать рекурсивный метод для вывода на экран следующей картинки:</p> <pre> 4 (1 раз) 555 (3 раза) 66666 (5 раз) ... (N раз) </pre>
13	<p>дано натуральное четное число n. Разработать рекурсивный метод для вывода на экран следующей картинки:</p> <pre> ***** (N символов) ***** (N-2 символа) *** ... * </pre> <p>Количество символов в первой строке введите с клавиатуры.</p>
14	<p>дано натуральное четное число n. Разработать рекурсивный метод для вывода на экран следующей картинки:</p> <pre> * * (между символами N пробелов) ** ** (между символами N-2 пробела) *** *** (между символами N -4 пробела) ... **** **** (между символами 2 пробела) ***** (между символами 0 пробелов) </pre>

Задание №2. Составить программу, содержащую рекурсивный метод СУММА_ЦИФР(K) целого типа, который находит сумму цифр целого числа K, не используя оператор цикла. С помощью этого метода найти суммы цифр для пяти целых чисел, введенных с клавиатуры.

НЕСТАТИЧЕСКИЕ МЕТОДЫ

Кроме статических методов (методов класса), в C# существуют нестатические методы (методы объектов). В заголовке нестатических методов указывается модификатор доступа. Нестатические методы реализуют функциональность объектов.

Модификатор	Назначение
static	метод доступен на уровне класса (не создавая объект (экземпляр) класса). Модификатор static используется для объявления статического члена, принадлежащего собственно типу, а не конкретному объекту. Модификатор static можно использовать с классами, полями, методами, свойствами, операторами, событиями и конструкторами, но нельзя — с индексаторами, методами завершения или типами, отличными от классов[4].
private - закрытый (по умолчанию)	Метод будет доступен только <i>внутри класса, где он определен</i> . Если при объявлении метода модификатор явно не указан, то используется по умолчанию модификатор private .
internal - внутренний доступ	Метод будет доступен из всех классов внутри сборки, в которой он определен . Внутренние типы или члены доступны только внутри <i>файлов</i> в одной и той же сборке. Из-за пределов этой сборки обратиться к нему будет нельзя.
protected - защищенный	Метод будет доступен только внутри самого класса и внутри любого производного от него класса . Для остальных вызовов из внешнего мира такой метод будет недоступен.
public - открытый	Модификатор общедоступности метода. К нему можно обратиться из любого метода любого класса программы.
protected internal - защищённый внутренний доступ	Доступен для самого класса и его наследников внутри сборки, где они определены. Доступ к типу или элементу может осуществляться любым кодом в сборке, в которой он объявлен, или из наследованного класса другой сборки. Доступ из другой сборки должен осуществляться в пределах объявления класса, производного от класса, в котором объявлен защищённый внутренний элемент, и должен происходить через экземпляр типа производного класса.
private protected - частный защищенный	Члены класса доступны только для классов-наследников, которые размещены в том же проекте, где определен исходный класс

Доступ к *нестатическим членам* класса, в том числе и к полям, методам осуществляется с помощью *объекта*. Объект создается **конструктором**.

Конструкторы – это специальные методы, которые используются для *инициализации объектов* при создании этих объектов.

Конструкторы разделяют:

- конструкторы "по умолчанию";
- конструкторы, созданные пользователем.

Модификатор доступа public

Пример 1. Описан нестатический метод SL1() с модификатором доступа public.

```
1. class Program
2. {
3.     public void SL1() //описание метода SL без параметров
4.     {
5.         int a = 1; int b = 2;
6.         int p = 2 * (a + b);
7.         Console.WriteLine("p={0}", p);
8.     }
9.     static void Main(string[] args)
10.    {
11.        Program ob = new Program(); // создание объекта ob для класса Program
12.        ob.SL1();// вызов метода SL1
13.    }
14. }
```

Строки 3 - 8: описан не статический метод SL1(). Строка 11. Для вызова нестатического метода необходимо создать объект класса Program. Объект ob создан с помощью конструктора "по умолчанию". Конструктор «по умолчанию» вызывается автоматически при создании объекта. Например, для класса Program –конструктор Program(). Строка 12 - вызов метода SL1(). Вызов метода осуществляется с использованием оператора «точка» : имя объекта.имя метода([параметры])

Пример 2

а) повторение. Использование статических методов SL_DROB_1 и РАЗНОСТЬ_ДРОБЕЙ для вычисления соответственно суммы дробей ($1/2 + 1/4$) и разности дробей ($1/2 - 1/4$).

namespace ConsoleApplication1

```
{
    class Program
    {
        static void SL_DROB_1(int a, int b, int c, int d)
        {
            double s = (double)a / b + (double)c / d;
            Console.WriteLine(" Сумма дробей =" + s);
        }
        static double РАЗНОСТЬ_ДРОБЕЙ(int a, int b, int c, int d)
        {
            // SL_DROB_1(a, b, c, d);
            double R = (double)a / b - (double)c / d;
            return R;
        }
        // =====
        static void Main(string[] args)
        {
            int a = 1, b = 2, c = 1, d = 4;
            SL_DROB_1(a, b, c, d); // вызов статического метода SL_DROB_1
            double R = РАЗНОСТЬ_ДРОБЕЙ(a, b, c, d); // вызов статического метода
            РАЗНОСТЬ_ДРОБЕЙ
            Console.WriteLine(" Разность дробей =" + R);
        }
    }
}
```

Результат работы программы:

Сумма дробей =0,75

Разность дробей =0,25

б) использование нестатических методов **SL_DROB_1** и **РАЗНОСТЬ_ДРОБЕЙ** для вычисления соответственно суммы дробей ($1/2 + 1/4$) и разности дробей ($1/2 - 1/4$).

```
namespace ConsoleApplication1
{
    class Program
    {
        public void SL_DROB_1(int a, int b, int c, int d)// нестатический метод SL_DROB_1
        {
            double s = (double)a / b + (double)c / d;
            Console.WriteLine(" сумма дробей =" + s);
        }

        public double РАЗНОСТЬ_ДРОБЕЙ(int a, int b, int c, int d)
        {
            // SL_DROB_1(a, b, c, d); // !! Вызов метода SL_DROB_1
            double R = (double)a / b - (double)c / d;
            return R;
        }
        // =====

        static void Main(string[] args)
        {
            int a = 1, b = 2, c = 1, d = 4;
            Program ob1 = new Program(); // Создание объекта для класса Program
            ob1.SL_DROB_1(a, b, c, d); // вызов нестатического метода SL_DROB_1 -
            применение к объекту данного метода

            Program ob2 = new Program();
            double R = ob2.РАЗНОСТЬ_ДРОБЕЙ(a, b, c, d);
            Console.WriteLine(" Разность дробей =" + R);
        }
    }
}
```

Результат работы программы:

Сумма дробей =0,75

Разность дробей =0,25

Модификаторы доступа: **private**, **internal**, **protected** для рассматриваемых методов, в пределах одного класса действуют аналогично.

Модификатор доступа private

Пример 3

Объединим методы **SL_DROB_1** и **РАЗНОСТЬ_ДРОБЕЙ** в отдельный класс **ДРОБЬ** и установим для метода **SL_DR** модификатор доступа **private**:

```

namespace ConsoleApplication1
{
    public class ДРОБЬ
    {
        // ----- сложение дробей -----
        private void SL_DR(int a, int b, int c, int d)
        {
            double s = (double)a / b + (double)c / d;
            Console.WriteLine(" сумма дробей =" + s);
        }
        //----- разность дробей -----
        public double ПАЗН_ДР(int a, int b, int c, int d)
        {
            //SL_DR(a, b, c, d);//2)–Вызов метода в пределах класса, в котором он создан
            double R = (double)a / b + (double)c / d;
            return R;
        }
        //----- умножение дробей -----
        // ... ....
    } // class ДРОБЬ

    class Program
    {
        static void Main(string[] args)
        {
            int a = 1, b = 2, c = 1, d = 4;
            ДРОБЬ ob2 = new ДРОБЬ(); // Создан объект класса ДРОБЬ
            ob2.SL_DR(a, b, c, d); // 1) - !! Нельзя вызвать метод из-за уровня доступа

            double R = ob2.ПАЗН_ДР(a, b, c, d);
            Console.WriteLine(" Разность дробей =" + R);
        }
    }
} // namespace ConsoleApplication1

```

Задания для выполнения в аудитории

Задание №1. Разработать программу, содержащую:

- Класс, представляющий треугольник. Предусмотреть в данном классе:
 - нестатические методы с параметрами для вычисления:
 - площади треугольника по формуле

$$S = \sqrt{p \times (p-a) \times (p-b) \times (p-c)}$$
, где p – полупериметр треугольника, a, b, c – длины сторон треугольника.

- периметра треугольника.

Модификаторы доступа для методов определите самостоятельно.

- Класс Program, содержащий:
 - поля, определяющие длины сторон треугольника;
 - Вызов методов для расчета площади и периметра.

Задания для самостоятельной работы студента (СРС)

Задание №1. Разработать программу, содержащую отдельный класс, в котором будут располагаться методы:

- Метод 1, вычисляющий факториал числа и возвращающий это значение. Метод 1 должен принадлежать только этому классу.
- Метод 2, определяющий сумму факториалов: $0!+1!+2!+\dots+N!$. Модификатор доступа к методу определите самостоятельно.

Написанная программа должна демонстрировать все разработанные элементы классов.

Задание №2. Во всех вариантах модификаторы доступа к методам должны обеспечивать доступность методов только в пределах сборки, в которой эти методы созданы.

Вариант	Задание
1	<p>Разработать программу, содержащую отдельный класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в <i>заданном</i> диапазоне. Счетчик имеет два метода:</p> <ul style="list-style-type: none"> • увеличения значения на единицу • уменьшения значения на единицу. <p>Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями, например вводом с клавиатуры.</p> <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
2	<p>Разработать программу, содержащую отдельный класс, реализующий шестнадцатеричный счетчик, который может увеличивать или уменьшать свое значение на единицу в <i>заданном</i> диапазоне. Счетчик имеет два метода:</p> <ul style="list-style-type: none"> • увеличения значения на единицу • уменьшения значения на единицу. <p>Предусмотреть инициализацию счетчика значениями по умолчанию.</p> <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
3	<p>Разработать программу, содержащую отдельный класс для многочлена вида $ax^2 + bx + c$. Предусмотреть методы, реализующие:</p> <ul style="list-style-type: none"> • вычисление значения многочлена для заданного аргумента; • операцию сложения, вычитания и умножения многочленов с получением нового объекта-многочлена; • вывод на экран описания многочлена. <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
4	<p>Разработать программу, содержащую отдельный класс, представляющий круг. Предусмотреть методы:</p> <ul style="list-style-type: none"> • для вычисления площади круга; • длины окружности; • проверки попадания заданной точки внутрь круга. <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>

Вариант	Задание
5	<p>Разработать программу, содержащую отдельный класс, представляющий прямоугольник. Предусмотреть методы:</p> <ul style="list-style-type: none"> • для вычисления площади прямоугольника; • периметра прямоугольника; • вычисления длины окружности, описанной около прямоугольника. <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
6	<p>Разработать программу, содержащую отдельный класс, реализующий двоичный счетчик, который может увеличивать или уменьшать свое значение на единицу в <i>заданном</i> диапазоне. Счетчик имеет два метода:</p> <ul style="list-style-type: none"> • увеличения значения на единицу; • уменьшения значения на единицу. <p>Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями, например вводом с клавиатуры.</p> <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
7	<p>Разработать программу, содержащую отдельный класс, представляющий ромб. Предусмотреть методы:</p> <ul style="list-style-type: none"> • для вычисления площади ромба по заданной стороне и синусу угла; • периметра ромба; • вычисления радиуса вписанной окружности; <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
8	<p>Разработать программу, содержащую отдельный класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в <i>заданном</i> диапазоне. Счетчик имеет два метода:</p> <ul style="list-style-type: none"> • увеличения значения на единицу; • уменьшения значения на единицу. <p>Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями, например вводом с клавиатуры.</p> <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
9	<p>Разработать программу, содержащую отдельный класс, реализующий шестнадцатеричный счетчик, который может увеличивать или уменьшать свое значение на единицу в <i>заданном</i> диапазоне. Счетчик имеет два метода:</p> <ul style="list-style-type: none"> • увеличения значения на единицу; • уменьшения значения на единицу. <p>Предусмотреть инициализацию счетчика значениями по умолчанию.</p> <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>

Вариант	Задание
10	<p>Разработать программу, содержащую отдельный класс для многочлена вида $ax^2 + Bx + c$. Предусмотреть методы, реализующие:</p> <ul style="list-style-type: none"> • вычисление значения многочлена для заданного аргумента; • операцию сложения, вычитания и умножения многочленов с получением нового объекта-многочлена; • вывод на экран описания многочлена. <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
11	<p>Разработать программу, содержащую отдельный класс, представляющий круг. Предусмотреть методы:</p> <ul style="list-style-type: none"> • для вычисления площади круга; • длины окружности; • проверки попадания заданной точки внутрь круга. <p>Написанная программа, должна демонстрировать все разработанные элементы классов.</p>
12	<p>Разработать программу, содержащую отдельный класс, представляющий прямоугольник. Предусмотреть методы:</p> <ul style="list-style-type: none"> • для вычисления площади прямоугольника; • периметра прямоугольника; • вычисления длины окружности, описанной около прямоугольника. <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
13	<p>Разработать программу, содержащую отдельный класс, реализующий двоичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Счетчик имеет два метода:</p> <ul style="list-style-type: none"> • увеличения значения на единицу; • уменьшения значения на единицу. <p>Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями, например вводом с клавиатуры.</p> <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>
14	<p>Разработать программу, содержащую отдельный класс, представляющий ромб. Предусмотреть методы:</p> <ul style="list-style-type: none"> • для вычисления площади ромба по заданной стороне и синусу угла; • периметра ромба; • вычисления радиуса вписанной окружности. <p>Написанная программа должна демонстрировать все разработанные элементы классов.</p>

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Кудрина Е.В., Огнева М.В., Портенко М.С. Программирование на языке C#: разработка консольных приложений. М.: Национальный открытый университет "ИНТУИТ", 2016.
2. Павловская Т.А. C#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
3. aspx - Рекурсивные функции: справочник по C#. URL: [https://msdn.microsoft.com/ru-ru/library/4bftz997\(v=vs.120\)](https://msdn.microsoft.com/ru-ru/library/4bftz997(v=vs.120)).
4. Out: справочник по C#. URL: [https://msdn.microsoft.com/ru-ru/library/t3c3bfhx\(v=vs.120\)](https://msdn.microsoft.com/ru-ru/library/t3c3bfhx(v=vs.120)).
5. Ref: справочник по C#. URL: [https://msdn.microsoft.com/ru-ru/library/14akc2c7\(v=vs.120\)](https://msdn.microsoft.com/ru-ru/library/14akc2c7(v=vs.120)).
6. Static: справочник по C#. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/static>.
7. Использование конструкторов: руководство по программированию на C#. URL: [https://msdn.microsoft.com/ru-ru/library/ms173115\(v=vs.120\)](https://msdn.microsoft.com/ru-ru/library/ms173115(v=vs.120)).
8. Методы: руководство по программированию на C#. URL: <https://msdn.microsoft.com/ru-ru/library/ms173114%28v=vs.120%29>.
9. Модификаторы доступа: руководство по программированию в C#. URL: [https://msdn.microsoft.com/ru-ru/library/ms173121\(v=vs.120\)](https://msdn.microsoft.com/ru-ru/library/ms173121(v=vs.120)).
10. Рекурсия - Википедия. Свободная энциклопедия. URL: <https://ru.wikipedia.org/wiki/>
11. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM
12. Статические классы и члены статических классов: руководство по программированию в C#. URL: [https://msdn.microsoft.com/ru-ru/library/79b3xss3\(v=vs.120\)](https://msdn.microsoft.com/ru-ru/library/79b3xss3(v=vs.120)).

Приложение 1

Таблица некоторых значений по умолчанию

Тип значения	Значение по умолчанию	Тип значения	Значение по умолчанию
bool	false	long	0 L
byte	0	sbyte	0
char	'\0'	short	0
decimal	0,0 M	uint	0
double	0,0 D	ulong	0
float	0,0 F	ushort	0
int	0	—	—

КОНСТРУКТОРЫ

I. СТАТИЧЕСКИЕ И НЕСТАТИЧЕСКИЕ ЧЛЕНЫ КЛАССА

Статические поля доступны всем методам класса.

Статические поля класса

Пример 1. Член класса, поле X объявлен с модификатором доступа static, следовательно, доступен на уровне класса.

```
namespace ConsoleApplication1
{
    class Program
    {
        static int X=2; // целочисленная переменная (поле) X
        static void Main(string[] args)
        {
            X=X+1; // доступ к статическому полю
            Console.WriteLine(" X = "+X);
        }
    }
}
```

Результат работы программы: X = 3

Пример 2.

Статическое целочисленное поле *a* (см. строку 1 программного кода) с модификатором доступа static, доступно для использования методами: t1 (см. строку 4, статический метод), t2 (см. строку 9, модификатор доступа к методу - public), методу Main(см. строку 14).

```
namespace ConsoleApplication1
{
    class Program
    {
        1. static int a = 1;
        2. static void t1()
        3. {
        4.     a = a + 200;
        5.     Console.WriteLine("Вывод в методе t1(static): a=" + a);
        6. }
        7. public void t2()
        8. {
        9.     a = a + 100;
        10. Console.WriteLine("Вывод в методе t2(public): a=" + a);
        11.}

        //-----
        12. static void Main(string[] args)
        13. {
        14. Console.WriteLine("Вывод в Main: a=" + a);
        15. t1();
        16. Program ob1 = new Program();
        17. ob1.t2();
        18. }
    }
}
```

Результат работы программного кода:

Вывод в Main: a=1
Вывод в методе t1(static): a=201
Вывод в методе t2(public): a=301

Статическое поле принадлежит самому классу и является общим для всех экземпляров (членов) этого класса.

Нестатические поля класса

Доступ к нестатическим членам класса, в том числе и к полям, осуществляется с помощью объекта. Объект создается **конструктором**.

Пример 3. Член класса, поле X (см. строка 1 программного кода), объявлен с модификатором доступа public, следовательно, поле доступно из любого другого кода в той же сборке(проекте) или из другой сборки(проекта), ссылающейся на него. Для доступа к нестатическому полю класса (см. строку 1, **public int X=2**), создается объект **ob** класса Program (см. строку 4 кода). Создание объекта **ob** осуществляется с помощью *конструктора* «по умолчанию», без параметров (**Program()**).

```
namespace ConsoleApplication1
{
    class Program
    {
        1. public int X; // 1) целочисленная переменная (поле) X инициализируется нулем
                       // 2) в классе конструктор не описан
        2. static void Main(string[] args)
        3. {
        4.     Program ob = new Program(); // создание объекта ob для класса Program
        5.     ob.X=ob.X+1; // доступ к нестатическому полю X через объект ob
        6.     Console.WriteLine(" X = "+ob.X);
        }
    }
}
```

Результат работы программы: X = 1

Примечание: модификаторы доступа не могут быть указаны для пространств имен; пространства имен не имеют ограничений доступа; типы верхнего уровня, не вложенные в другие типы, могут иметь только уровень доступности *internal* или *public*, для таких типов уровнем доступности по умолчанию является *internal*.

II. СПЕЦИАЛЬНЫЕ МЕТОДЫ – КОНСТРУКТОРЫ

Конструкторы – это специальные методы, которые используются для инициализации объектов при их создании.

Конструктор – это метод класса, который вызывается для создания объекта этого класса.

Конструктором называется группировка кода, которой передаётся управление при создании объекта.

Синтаксис описания класса:

[атрибуты][спецификаторы]class_имя_класса[:список_родителей]

```

{ //тело_класса
  [Статические поля - константы];
  [поля];
  [конструкторы];
  [деструкторы];
  [индексаторы];
  [методы];
  [события];
  [классы (структуры, делегаты, интерфейсы, перечисления)].
}

```

Если конструктор не описан в классе, то запускается конструктор по умолчанию. Для создания объекта можно описать свой конструктор (конструктор пользователя). Конструктор пользователя часто применяется в тех случаях, когда необходима инициализация объектов не нулевыми значениями.

Синтаксис объявления конструктора аналогичен объявлению метода.

Синтаксис описания конструктора:

```

[атрибуты] имя_класса ([параметры])
{
  тело конструктора
}

```

Пример 4. Создание объекта класса Random с помощью конструктора «по умолчанию».

```

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            //Создание объекта класса Random ( Random входит в пространство имен
            System)
            Random rnd; rnd = new Random(); // Random rnd = new Random();

            Console.WriteLine("rnd=" + rnd.NextDouble());
            Console.WriteLine("rnd=" + rnd.Next(14));
        }
    }
}

```

Пример 5. Создание объекта класса Program с помощью конструктора пользователя без параметров.

Для доступа и инициализации нестатического поля X (см. строку 1) класса Program, используется описанный пользователем конструктор без параметров (см. строки 2-5). В строке 8 с помощью описанного конструктора создается объект **ob** для вывода на экран значения поля X (см. строку 9).

```

namespace ConsoleApplication1
{
    class Program
    {
        1. public int X;
        2. public Program()//конструктор без параметров
        3. {
        4.     X = 5; // зн-е X инициализировано в конструкторе
        5. }

        // =====
        6. static void Main(string[] args)
        7. {
        8.     Program ob = new Program(); // создание объекта ob для класса Program
        9.     Console.WriteLine(" X = " + ob.X);
        10. }
    }
}

```

Результат работы программы: X = 5.

Пример 6. Объект создается конструктором «по умолчанию». Данный конструктор инициализирует значения полей объекта sa, sb; нулями. В методе М периметр вычисляется: $PR = 2 * (sa + sb) = 2*(0+0)$.

```

namespace ConsoleApplication1
{
    class Rect
    {
        // sa,sb - стороны прямоугольника
        public int sa, sb;
        public double M()
        {
            double PR = 2 * (sa + sb); // sa,sb принимают значения 2 и 3 соответственно
            return PR;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            //Создание объекта класса Rect из пространства имен ConsoleApplication1
            // Вызов конструктора «по умолчанию»
            Rect pr1 = new Rect();
            Console.WriteLine("sa=" + pr1.sa + " sb=" + pr1.sb);
            Console.WriteLine("Периметр =" + pr1.M());
        }
    }
}

```

Результат работы программы:

sa=0 sb=0

Периметр =0

Добавим конструктор пользователя без параметров (см. строки 2-5) для инициализации сторон прямоугольника не нулевыми значениями, а указанными пользователем значениями (см. строку 4).


```

namespace ConsoleApplication1
{
    class Rect
    {
        // sa,sb - стороны прямоугольника
        1. public int sa, sb;
        2. public Rect( )
        3. {
        4.     sa = 10; sb = 20;
        5. }
        public double M()
        {
            double PR = 2 * (sa + sb); // sa,sb принимают значения 2 и 3 соответственно
            return PR;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            //Создание объекта класса Rect из пространства имен ConsoleApplication1
            Rect pr1 = new Rect(); // Вызов конструктора пользователя без параметров
            Console.WriteLine("sa=" + pr1.sa + " sb=" + pr1.sb);
            Console.WriteLine("Периметр =" + pr1.M());
        }
    }
}

```

Результат работы программы:

sa=10 sb=20

Периметр =60

Пример 7. Создание объекта класса **Rect** с помощью *конструктора пользователя с параметрами*.

Инициализация полей **sa** и **sb** осуществляется с помощью конструктора пользователя (см. строки 2-5) с параметрами 2 и 3 .

```
namespace ConsoleApplication1
{
    class Rect
    {
        // sa,sb - стороны прямоугольника
        1. public int sa, sb;
        // Конструктор с параметрами: инициализация полей класса (sa,sb)
        2. public Rect(int a, int b)
        3. {
        4.     sa = a; sb = b;
        //Console.WriteLine("sa=" + sa);
        5. }
    }

    class Program
    {
        static void Main(string[] args)
        {
            //Создание объекта класса Rect из пространства имен ConsoleApplication1
            // Вызов конструктора с параметрами (2, 3):
            Rect pr1 = new Rect(2, 3);
            Console.WriteLine("sa=" + pr1.sa + " sb=" + pr1.sb);
        }
    }
}
```

Результат работы программы:

sa=2 sb=3

Пример 8. Вычисление периметра прямоугольника (метод M(), см. строки 6-10), стороны которого инициализированы конструктором пользователя с параметрами (см. строку 16).

```
namespace ConsoleApplication1
```

```
{
    class Rect
    {
        // sa,sb - стороны прямоугольника
        1. public int sa, sb;
        // Конструктор 1: инициализация полей класса (sa,sb)
        2. public Rect(int a, int b)
        3. {
        4.     sa = a; sb = b;
        5. }
        // Метод вычисления периметра прямоугольника
        6. public double M()
        7. {
        8.     double PR = 2 * (sa + sb); // sa,sb принимают значения 2 и 3 соответственно
        9.     return PR;
        10. }
        11. }

    class Program
    {
        static void Main(string[] args)
        {
            //Создание объекта класса Rect из пространства имен ConsoleApplication1

            16. Rect pr1 = new Rect(2, 3); // Вызов конструктора с параметрами (2,3):
            17. Console.WriteLine("sa=" + pr1.sa + " sb=" + pr1.sb);
            18. Console.WriteLine("Периметр =" + pr1.M()); // вызов метода
        }
    }
}
```

Результат работы программы:

A screenshot of a console window with a black background and white text. The text shows the output of the program: "sa=2 sb=3" on the first line, "Периметр =10" on the second line, and "Для продолжения нажмите любую клавишу . . ." on the third line.

Задания для выполнения в аудитории

Задание №1. Описать отдельный класс, представляющий треугольник.

Предусмотреть:

- поля класса: стороны треугольника;
- методы для вычисления: площади, периметра;
- два конструктора пользователя (один с параметрами, второй без параметров) для инициализации объектов (например: равносторонний, равнобедренный, прямоугольный треугольник);

Разработать программу, демонстрирующую все разработанные элементы класса.

Задания для самостоятельной работы студента (СРС)

Задание №1. Выполните задание, согласно Вашему варианту.

Вариант	Задание
1	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй с параметрами только вещественного типа), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: длины вектора для пространственных задач $ a = \sqrt{a_x^2 + a_y^2 + a_z^2}$.
2	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй только с целочисленными параметрами), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: скалярного произведения двух векторов для пространственных задач: $\vec{a} \cdot \vec{b} = a_x \cdot b_x + a_y \cdot b_y + a_z \cdot b_z$.
3	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй с параметрами только вещественного типа), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: сложения двух векторов, с получением нового вектора для пространственных задач: $\vec{a} + \vec{b} = \{a_x + b_x; a_y + b_y; a_z + b_z\}$.
4	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй только с целочисленными параметрами), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: разности двух векторов, с получением нового вектора для пространственных задач: $\vec{a} - \vec{b} = \{a_x - b_x; a_y - b_y; a_z - b_z\}$.
5	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй только с целочисленными параметрами), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: длины вектора для пространственных задач $ a = \sqrt{a_x^2 + a_y^2 + a_z^2}$.

Вариант	Задание
6	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй только с вещественными параметрами), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: разности двух векторов, с получением нового вектора для пространственных задач: $\vec{a} - \vec{b} = \{a_x - b_x; a_y - b_y; a_z - b_z\}$.
7	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй только с вещественными параметрами), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: скалярного произведения двух векторов для пространственных задач: $\vec{a} \cdot \vec{b} = a_x \cdot b_x + a_y \cdot b_y + a_z \cdot b_z$.
8	Составьте описание класса «Вектор» для создания вектора, заданного координатами начальной и конечной точек. Определите в составе класса: поля (координаты точек), конструкторы (один конструктор без параметров, второй только с целочисленными параметрами), необходимые методы. Создайте объекты для работы с классом. Обеспечьте следующие операции вычисления: сложения двух векторов, с получением нового вектора для пространственных задач: $\vec{a} + \vec{b} = \{a_x + b_x; a_y + b_y; a_z + b_z\}$.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
2. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM
3. Конструкторы: руководство по программированию на C#. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/constructors>

СВОЙСТВА

Свойство — это член класса, предоставляющий гибкий механизм для чтения, записи или вычисления значения частного поля. Свойства представляют собой специальные методы, - *методы доступа к данным*, обеспечивают организацию доступа к полям класса.

Общая форма свойства:

```
тип имя
{
    set
    {
        // код аксессора для записи в поле
    }
    get
    {
        // код аксессора для чтения из поля
    }
}
```

Метод доступа *set* используется для присвоения (установки) нового значения. Метод доступа *get* используется для возврата (получения) значения свойства. Ключевое слово *value* используется для определения значения, присваиваемого методом доступа *set*. Может отсутствовать один из методов *set* или *get*.

Пример 1. В методе *set* используется *ключевое слово value*
namespace ConsoleApplication1

```
{
    class Program
    {
        int pol; //закрытое поле pol; «начальное» значение=0
        public int Pol // Pol - св-во поля pol
        {
            set
            {
                pol = value; // 3) Value=5 – хранит измененное значение поля pol,=> pol=5;
            }
            get
            {
                return pol; // 5) возврат поля pol=5
            }
        }
    }
}
... ..
static void Main(string[] args)
{
    Program ob = new Program();

    ob.Pol=5; // 1) ob.Pol - слева в выражении ob.Pol=5 => set; 2) Value =5
    int rez= ob.Pol ; // 4) ob.Pol – справа в выражении => get;
    Console.WriteLine("rez =" + rez); //6) rez=5
}
} // class Program
} // namespace ConsoleApplication1
```

Пример 2. Использование свойств: Контроль значений переменной A: A – должна быть только положительным значением.

```
namespace ConsoleApplication9
{
    class Program
    {
        int A;
        public int SV_A
        {
            get
            {
                return A;
            }
            set
            {
                if (value >= 0) A = value;
                else A=100;
            }
        }
    }

    static void Main(string[] args)
    {
        Program ob = new Program();
        ob.SV_A = -2;
        Console.WriteLine("SV_A =" + ob.SV_A);

        ob.SV_A = 200;
        Console.WriteLine("SV_A =" + ob.SV_A);

        Console.ReadLine();
    }
}
```

Результат работы программы:

```
SV_A =1000
SV_A =200
```

Пример 3. Несколько свойств

Дано:

Класс **Persona**, у которого четыре *поля*:

`string fam = ""`, `status = ""`; `int age = 0`, `zarplata = 0`;

Все поля закрыты.

- **fam** – фамилия. Стратегия доступа – можно задать только один раз: использование аксессоров set и get (присваивать полю fam значение, если Fam == "", например, аксессор set:
`set { if (fam == "") fam = value; }`, аксессор get запишите самостоятельно).

- **status** – «ребенок», если возраст <7; «школьник», если возраст >=7, но <17; «студент», если возраст >=17 но <=22; «служащий», если возраст > 22 лет.
Стратегия доступа - только для чтения.
В свойство поля status информация передается из свойства поля age (возраст).
- **age** – возраст. Стратегия доступа – для чтения и записи
- **zarplata** – зарплата. Стратегия доступа – для записи

Реализация класса Persona:

```
public class Persona
```

```
{
```

```
    string fam = "", status = "";
```

```
    int age = 0, zarplata = 0;
```

```
    //стратегия: Чтение, запись при первом обращении
```

```
    public string Fam
```

```
    {
```

```
        set { if (fam == "") fam = value; }
```

```
        get { return fam; }
```

```
    }
```

```
    //стратегия: Только чтение
```

```
    public string Status
```

```
    {
```

```
        get { return status; }
```

```
    }
```

```
    //стратегия: Чтение, запись
```

```
    public int Age
```

```
    {
```

```
        set
```

```
        {
```

```
            age = value;
```

```
            if (age < 7) status = "ребенок";
```

```
            else if (age < 17) status = "школьник";
```

```
            else if (age < 22) status = "студент";
```

```
            else status = "служащий";
```

```
            if (value < 7) status = "ребенок";
```

```
            else if (value < 17) status = "школьник";
```

```
            else if (value < 22) status = "студент";
```

```
            else status = "служащий";
```

```
            age = value;
```

```
        }
```

```
        get { return age; }
```

```
    }
```

```
    //стратегия: Только запись
```

```
    public int Zarplata
```

```
    {
```

```
        set { zarplata = value; }
```

```
    }
```

```
}
```

Задания для выполнения в аудитории

Задание №1

1. Исходя из данных Примера 3, составьте программу, которая будет использовать свойства: Fam, Age, Status, Zarplata, соответствующих полей.

Присвоив им определенные значения, выведите на экран:

- фамилию;
- возраст;
- статус, соответствующий возрасту.

Например, в результате работы программы на экране должны отображаться следующие поля, например:

Фамилия = Петров, возраст =21, статус = студент

Для этого добавьте в программу класс **Program** и обеспечьте доступ ко всем разработанным элементам класса.

2. Дополните программный код для вывода на экран фамилии, возраста, статуса (в зависимости от возраста), зарплаты(в зависимости от статуса). Например:

Фамилия = Петров, возраст =35, статус = служащий, зарплата=20000.

Задания для самостоятельной работы студента (СРС)

Задание №1. Для всех вариантов при выполнении задания разрабатываются *отдельные классы* (помимо основного класса Program).

Вариант	Задание
1	<p>Необходимо описать класс «Товар», содержащий следующие закрытые поля:</p> <ul style="list-style-type: none"> • название товара; • название магазина, в котором продается товар; • стоимость товара в рублях. <p>Предусмотреть свойства для получения состояния объекта. Остальные элементы класса определите самостоятельно. Написать программу, демонстрирующую все разработанные элементы классов.</p>
2	<p>Необходимо описать класс, представляющий треугольник. В качестве полей используйте координаты точек для рассмотрения треугольника на плоскости. Предусмотреть свойства полей для размещения треугольника в первой координатной плоскости. Предусмотреть конструкторы с параметрами для инициализации объектов «треугольник равносторонний»(по координатам). Остальные элементы класса определите самостоятельно. Написать программу, демонстрирующую все разработанные элементы класса.</p>
3	<p>Необходимо описать класс «Поезд», который должен содержать следующие закрытые поля:</p> <ul style="list-style-type: none"> • номер поезда (целочисленное значение); • название пункта отправления; • название пункта назначения; • количество вагонов(целочисленное значение); • время прибытия; • время отправления. <p>Предусмотреть свойства для каждого поля с целью получения состояния объекта. Остальные элементы класса определите самостоятельно. Составить программу с обеспечением доступа ко всем разработанным элементам класса.</p>
4	<p>Необходимо описать класс, представляющий треугольник. В качестве полей используйте координаты точек для рассмотрения треугольника на плоскости. Предусмотреть свойства полей для размещения треугольника во второй координатной плоскости. Предусмотреть конструкторы с параметрами для инициализации объектов «треугольник равнобедренный»(по координатам). Остальные элементы класса определите самостоятельно. Написать программу, демонстрирующую все разработанные элементы класса.</p>
5	<p>Необходимо описать класс, представляющий треугольник. В качестве полей используйте координаты точек для рассмотрения треугольника на плоскости.</p>

Вари- ант	Задание
	<p>Предусмотреть свойства полей для размещения треугольника в третьей координатной плоскости.</p> <p>Предусмотреть конструкторы с параметрами для инициализации объектов «треугольник прямоугольный».</p> <p>Остальные элементы класса определите самостоятельно.</p> <p>Написать программу, демонстрирующую все разработанные элементы класса.</p>
6	<p>Создайте класс «Телевизор». Среди других полей (разрешение экрана, скорость интернет - соединения, диагональ), объявите в классе поле «громкость звука», для доступа к этому полю реализуйте свойство. Громкость требуется в диапазоне от 0 до 100 условных единиц измерения; разрешение экрана от 1080p до 2160p; скорость интернет-соединения для стандартного разрешения и комфортного просмотра онлайн-видео от 1 до 2 Мбит/с; диагональ 40-42 дюйма.</p>
7	<p>Необходимо описать класс, представляющий круг.</p> <p>В качестве полей используйте координаты центра круга.</p> <p>Предусмотреть свойства полей для размещения круга в четвертой координатной плоскости.</p> <p>Предусмотреть конструктор с параметрами для инициализации объекта «круг».</p> <p>Остальные элементы класса определите самостоятельно.</p> <p>Написать программу, демонстрирующую все разработанные элементы класса.</p>
8	<p>Необходимо описать класс «Товар», содержащий следующие закрытые поля:</p> <ul style="list-style-type: none"> • название товара; • название магазина, в котором продается товар; • стоимость товара в рублях. <p>Предусмотреть свойства для получения состояния объекта.</p> <p>Остальные элементы класса определите самостоятельно.</p> <p>Написать программу, демонстрирующую все разработанные элементы классов.</p>
9	<p>Необходимо описать класс, представляющий треугольник.</p> <p>В качестве полей используйте координаты точек для рассмотрения треугольника на плоскости.</p> <p>Предусмотреть свойства полей для размещения треугольника в первой координатной плоскости.</p> <p>Предусмотреть конструкторы с параметрами для инициализации объектов «треугольник равносторонний»(по координатам).</p> <p>Остальные элементы класса определите самостоятельно.</p> <p>Написать программу, демонстрирующую все разработанные элементы класса.</p>
10	<p>Необходимо описать класс «Поезд», который должен содержать следующие закрытые поля:</p> <ul style="list-style-type: none"> • номер поезда (целочисленное значение); • название пункта отправления; • название пункта назначения; • количество вагонов(целочисленное значение); • время прибытия; • время отправления. <p>Предусмотреть свойства для каждого поля с целью получения состояния объекта.</p>

Вариант	Задание
	Остальные элементы класса определите самостоятельно. Составить программу с обеспечением доступа ко всем разработанным элементам класса.
11	Необходимо описать класс, представляющий треугольник. В качестве полей используйте координаты точек для рассмотрения треугольника на плоскости. Предусмотреть свойства полей для размещения треугольника во второй координатной плоскости. Предусмотреть конструкторы с параметрами для инициализации объектов «треугольник равнобедренный»(по координатам). Остальные элементы класса определите самостоятельно. Написать программу, демонстрирующую все разработанные элементы класса.
12	Необходимо описать класс, представляющий треугольник. В качестве полей используйте координаты точек для рассмотрения треугольника на плоскости. Предусмотреть свойства полей для размещения треугольника в третьей координатной плоскости. Предусмотреть конструкторы с параметрами для инициализации объектов «треугольник прямоугольный». Остальные элементы класса определите самостоятельно. Написать программу, демонстрирующую все разработанные элементы класса.
13	Создайте класс «Телевизор». Среди других полей (разрешение экрана, скорость интернет - соединения, диагональ), объявите в классе поле «громкость звука», для доступа к этому полю реализуйте свойство. Громкость требуется в диапазоне от 0 до 100 условных единиц измерения; разрешение экрана от 1080p до 2160p; скорость интернет - соединения для стандартного разрешения и комфортного просмотра онлайн - видео от 1 до 2 Мбит/с; диагональ 40-42 дюйма.
14	Необходимо описать класс, представляющий круг. В качестве полей используйте координаты центра круга. Предусмотреть свойства полей для размещения круга в четвертой координатной плоскости. Предусмотреть конструктор с параметрами для инициализации объекта «круг». Остальные элементы класса определите самостоятельно. Написать программу, демонстрирующую все разработанные элементы класса.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
2. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM
3. Свойства: руководство по программированию на C#. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/properties>.

МАССИВЫ

ОДНОМЕРНЫЕ МАССИВЫ

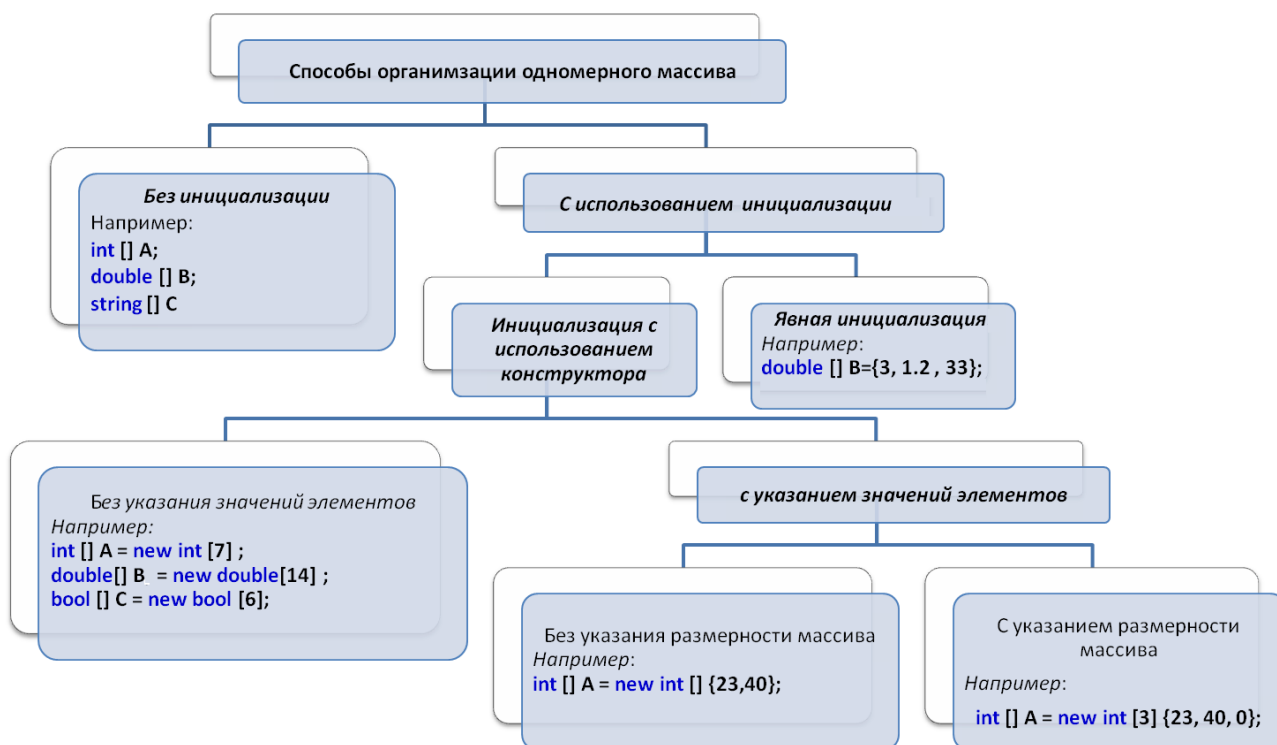
Одномерный массив представляется как список связанных переменных.

Для использования массива в программе необходимо:

1. Объявить переменную, которая может обращаться к массиву.
2. Создать экземпляр массива, используя оператор `new`.

Индекс первого элемента массива – нулевой. Если границы массива не достигаются или же превышаются, то возникает ошибка при выполнении.

Элементам массива присваиваются начальные значения в зависимости от базового типа. Для арифметических типов – нули, для ссылочных типов – `null`, для символов – символ с кодом ноль.



СПОСОБЫ ОРГАНИЗАЦИИ МАССИВА

1. Без инициализации (объявление с отложенной инициализацией)

При объявлении с отложенной инициализацией сам массив не формируется, а создается только ссылка на массив, имеющая неопределенное значение `Null`.

Например, объявления трех массивов с отложенной инициализацией: `int[] a, b, c;`

2. Объявление массива с инициализацией.

2.1. Инициализация массива при его создании (явная инициализация)

Инициализация является явной и задается константным массивом. Элементы константного массива заключаются в фигурные скобки. В памяти создается константный массив, с которым и связывается ссылка.

Общая форма инициализации одномерного массива:

`тип[] имя_массива = {v1, v2, v3, ..., vN} ;`

Пример 1. Объявление массива с явной инициализацией.

1.1. Инициализация целочисленного массива W:

```
int[] W = { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 11, 12 };
```

1.2. Инициализация массива вещественных чисел W1

```
double [] W1={110.7, -22.2, 0.2, 123.4};
```

2.2. Инициализация массива в объектном стиле (инициализация массива с использованием конструктора)

Общая форма для объявления одномерного массива:

```
тип[] имя_массива = new тип[размер] ;
```

тип – объявляет конкретный тип элемента массива;

[] – стоят после типа и указывают на объявление одномерного массива;

размер – определяет число элементов массива.

Пример 2. Объявление трех целочисленных одномерных массива A, B, C

```
int[] A = new int[18]; B = new int[4]; C = new int[25];
```

Тип `int(int[]...)` распространяется не только на массив A, но и на массивы B и C.

Пример 3. Объявление целочисленного одномерного массива A; массивы B, C не объявлены

```
int[] A = new int[18]; B = new int[4]; C = new int[25];
```

Пример 4. Создание целочисленного массива pr, состоящего из 12 элементов

1. Объявление целочисленного массива pr:

`int[] pr = new int[12];` – в переменной pr хранится ссылка на область памяти, выделяемой для массива оператором new;

или:

`int[] pr;` – переменная pr не ссылается на какой-то определенный физический объект;

`pr = new int[12];` - после выполнения данного оператора переменная pr будет ссылаться на массив; pr – переменная ссылки на массив, в ней хранится ссылка на область памяти, выделяемой для массива. Массив pr состоит из 12 элементов с индексами от 0 до 11.

ДЕЙСТВИЯ НАД ЭЛЕМЕНТАМИ МАССИВА

1. Заполнение массива.
2. Вывод элементов массива на экран.
3. Поиск в массиве.
4. Передача массива, как параметра, в метод.

1. Заполнение массива

- 1.1. Вводом с клавиатуры.
- 1.2. «Случайным образом».
- 1.3. По формуле.
- 1.4. Из текстового файла.

1.1. Вводом с клавиатуры

Фрагмент программного кода для ввода пяти элементов массива:

```
for (int i = 0; i < 5; i = i + 1)
{
    Console.WriteLine("Введите " + i + " элемент массива ");
    pr[i] = Convert.ToInt32(Console.ReadLine());
}
```

1.2. «Случайным образом»

Заполнение массива «случайными числами» осуществляется с помощью генератора случайных чисел. Для генерации произвольных чисел используется класс Random (пространство имён System) генератора псевдослучайных значений.

Технология использования класса Random:

1. создать объект класса Random;
2. для созданного объекта вызвать необходимый метод.

Наиболее часто используемые *методы* класса *Random*:

- int Next() – возвращает очередное псевдослучайное целое число в диапазоне от 0 до 0x7FFFFFFF;
- int Next(int Max) – то же в диапазоне от 0 до max;
- int Next(int Min, int Max) – в диапазоне от min до max;
- double NextDouble() – возвращает очередное псевдослучайное вещественное число в диапазоне от 0,0 до 1,0.

Пример 5. Заполнение массива случайными числами.

1. Заполнение массива целочисленными значениями

Фрагмент программного кода:

```
int[] massiv = new int[5];
Random R = new Random(); // инициализация генератора случайных чисел (создание
                          // объекта R для класса Random )
for (int i = 0; i < 4; i++) massiv[i] = R.Next(3, 15); // генерация случайного числа целого
                                                       // типа в диапазоне [3, 15] (для объекта R
                                                       // вызывается метод Next класса Random). //
```

Заполнение массива massiv сгенерированным числом.

2. Заполнение массива вещественными значениями

```
Random R = new Random(); // инициализация генератора случайных чисел (создание
                          // объекта R для класса Random )
double[] massiv = new double[5];
for (int i = 0; i < 4; i++) massiv[i] = R.NextDouble(); // Заполнение массива massiv
                                                         // вещественными числами в
                                                         // диапазоне от 0,0 до 1,0
```

1.3. По формуле

Заполнение одномерного массива *pr* по формуле $i * 2$.

Фрагмент программного кода:

```
int[] pr = new int[12];
for (int i = 0; i < 12; i = i + 1) pr[i] = i * 2;
```

1.4. Из текстового файла

Текстовые файлы позволяют выполнять только последовательный доступ.

Файлы с последовательным доступом – это файлы, хранящие информацию в неструктурированном виде, например файлы с расширением *.txt. При чтении данных из файла с последовательным доступом, данные считываются от начала к концу файла.

Преобразование из внутренней формы представления числа в символьную, выполняется с помощью перегруженных методов ToString, результаты выполнения которых передаются в методы текстовых файлов.

Для работы с текстовыми файлами используются классы из пространства имен System.IO. Например, *StreamWriter* и *StreamReader* – символьные потоки, которые работают с Юникодом.

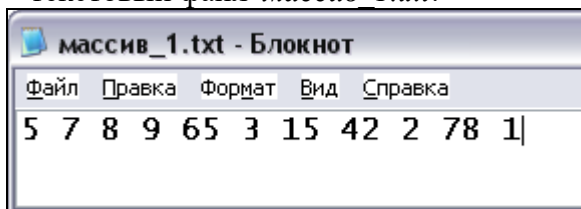
Перед работой с файлами необходимо добавить пространство имен System.IO:

```
using System;
using System.Linq;
using System.Text;
using System.IO;
```

Пример 6. Заполнение целочисленного одномерного массива *pr* значениями из текстового файла *массив_1.txt*

Дано:

– текстовый файл *массив_1.txt*:



Элементы файла записаны в одну строку, через пробел.

– описан целочисленный одномерный массив *pr*.

Требуется: заполнить массив *pr* из файла *массив_1.txt*

Решение:

1. Чтение из файла в строковую переменную:

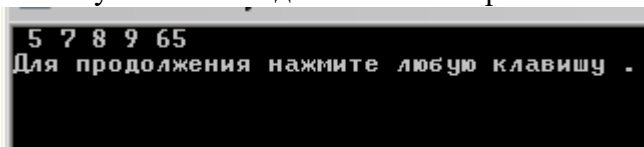
- создать объект, например, fl класса *StreamReader*:
`StreamReader fl = new StreamReader("D:/массив_1.txt");`
- прочитать из файла символы как одну строку в строковую переменную S (метод *ReadToEnd()*):
`string S=fl.ReadToEnd();`

Результат: в переменной S хранятся символы 5 7 8 9 65 3 15 42 2 78 1.

- Создается *строковый массив* s1 из символов строки S, с помощью метода *Split(' ')*. Метод *Split()* разделяет заданную строку на подстроки, в качестве разделителя использует символ, указанный в качестве параметра.
`string[] s1 = s.Split(' ');`
- Преобразование строкового массива s1 в целочисленный массив pr:

```
for (int j = 0; j < pr.Length; j++)
{
    pr[j] = Convert.ToInt32(s1[j]);
}
```

Результат: вывод массива на экран:



- Закрывать файл `fl.Close();`

2. Вывод элементов массива на экран

- с использованием цикла *for*
Пример 7. Вывод на экран массива pr с использованием цикла *for*

Фрагмент программного кода для вывода на экран целочисленного массива pr:

```
static void Main()
{
    int[] pr = new int[12];
    for (int i = 0; i < 12; i = i + 1) pr[i] = i * 2;
    for (int i = 0; i < 12; i++) Console.Write(" " + pr[i]);
}
```

Фрагмент программного кода для вывода на экран вещественных элементов массива mas, с тремя знаками после запятой:

```
double[] mas = new double[5];
for (int i = 0; i < N; i = i + 1) mas[i] = R.NextDouble() + R.Next(1,5);
for (int i = 0; i < N; i++) Console.Write("{0,7:N3}", mas[i]); // 7 символов на вывод числа,
включая запятую; N3 – три знака из 7 - на вывод дробной части
```

Результат вывода:

1,382 4,495 4,349 1,580 4,636

- с использованием цикла *foreach*:

При использовании оператора `foreach` необходимо указать элементы какой группы необходимо перебрать.

Синтаксис оператора:

`foreach (<тип> <имя> in <группа>) <тело цикла>;`

тип – соответствует базовому типу элементов *группы* (массива);

имя – определяет переменную, с помощью которой осуществляется перебор по очереди всех значений из указанной *группы* (массива).

Ограничением оператора `foreach`: с его помощью можно только *просматривать* значения элементов *группы* данных (массива).

Пример 8. Вывод на экран массива `pr` с помощью цикла `foreach`.

Фрагмент программного кода:

```
static void Main()
{
    int[] pr = new int[12];
    for (int i = 0; i < 12; i = i + 1) pr[i] = i * 2;
    foreach (int el in pr) Console.WriteLine(el);
}
```

3. Поиск в массиве

Пример 9. Поиск элемента с заданным значением в массиве *простым перебором*.

Фрагмент программного кода:

```
static void Main()
{
    int[] pr = new int[12];
    for (int i = 0; i < 12; i = i + 1) pr[i] = i * 2;
    for (int i = 0; i < 12; i = i + 1)
        if (pr[i] == 20) Console.Write(pr[i] + " "); // если элемент со значением 20 найден, то
                                                    // его значение выводится на экран
}
```

4. Передача массива, как параметра, в метод

Массив передается в метод по ссылке. Все изменения элементов массива, являющегося формальным параметром, отражаются на элементах соответствующего массива, являющегося фактическим параметром.

При передаче массива по ссылке `ref`, инициализация массива (как параметра) перед передачей – обязательна. При этом указывать спецификатор `ref` не обязательно.

Пример 10. Передача массива `PR` в метод `TR`, как параметра по ссылке `ref`.

Описание метода `TR` с параметром «массив»:

```
static void TR (int [] PR)
{
    //Тело метода
}
```

... ..

Обращение к методу `TR` с параметром «массив»:

```
int[] PR = { 100, 200, 300, 400 };
TR(PR);
```

Пример 11. Передача массива в метод RMAС как параметра по ссылке **ref**. В методе RMAС элементы массива увеличиваются на 100, после чего передаются в основной метод (return(A)).

Описание метода RMAС:

```
static int[] RMAС (ref int[] A)
{
    for(int i=0;i<4;i++) A[i]=A[i]+100;
    return (A);
}
```

..

Обращение к методу RMAС:

```
int[] A={ 10, 11, 12, 13 };
A=RMAС ( ref A);
```

Или:

Описание метода RMAС с параметром «массив» , с использованием класса **Array**:

```
static Array RMAС(ref int[] A)
{
    for(int i=0;i<4;i++) A[i]=A[i]+100;
    return (A);
}
```

..

Обращение к методу RMAС:

```
int[] A={ 10, 11, 12, 13 };
RMAС ( ref A);
```

Пример 12. Передачи массива в метод RMAС как параметра по ссылке **out**.

Описание метода RMAС с параметром «массив», тип метода **int**:

```
static int[] RMAС(out int[] A)
{
    A = new int[4] { 10, 11, 12, 13 };
    return (A);
}
```

..

Обращение к методу RMAС:

```
int[] A;
A=RMAС (out A);
```

Или:

Описание метода RMAС с параметром «массив», тип метода **void** :

```
static void RMAС(out int[] A)
{
    A = new int[4] { 10, 11, 12, 13 };
}
```

..

Обращение к методу RMAС:

```
int[] A;
RMAС (out A);
```

Или:

Описание метода RMAС с параметром «массив», с использованием класса **Array** :

```
static Array RMAS(out int[] A)
{
    A = new int[4] { 100, 110, 120, 130 };
    return (A);
}
```

.....

Обращение к методу RMAS:

```
int[] A;
RMAS (out A);
```

Некоторые свойства и методы, класса Array пространства имен System:

№	Элемент	Вид	Описание
1	<i>Length</i>	свойство	Количество элементов массива (по всем размерностям)
2	<i>BinarySearch()</i>	статический метод	Двоичный поиск в отсортированном массиве
3	<i>Clear()</i>	статический метод	Присваивание элементам массива значений по умолчанию (0 для арифметического, <i>false</i> для логического, <i>null</i> для ссылочного типа)
4	<i>Copy()</i>	статический метод	Копирование заданного диапазона элементов одного массива в другой
5	<i>CopyTo()</i>	экземплярный метод	Копирование всех элементов текущего одномерного массива в другой массив
6	<i>Find()</i>	статический метод	Поиск первого элемента, удовлетворяющего заданному условию
7	<i>FindAll()</i>	статический метод	Возвращает одномерный массив элементов, удовлетворяющих заданному условию
8	<i>FindIndex()</i>	статический метод	Возвращает индекс первого вхождения элемента, удовлетворяющего заданному условию
9	<i>FindLast()</i>	статический метод	Возвращает последнее вхождение элемента, удовлетворяющего заданному условию
10	<i>FindLastIndex()</i>	статический метод	Возвращает индекс последнего вхождения элемента, удовлетворяющего заданному условию
	<i>GetLength()</i>	экземплярный метод	Количество элементов в заданном измерении массива
12	<i>GetValue()</i>	экземплярный метод	Получение значения элемента массива
13	<i>IndexOf()</i>	статический метод	Поиск первого вхождения элемента в одномерный массив
14	<i>LastIndexOf()</i>	статический метод	Поиск последнего вхождения элемента в одномерный массив
15	<i>Rank</i>	свойство	Количество измерений массива
16	<i>Reverse()</i>	статический метод	Изменение порядка следования элементов на обратный
17	<i>SetValue()</i>	экземплярный метод	Установка значения элемента массива
18	<i>Sort()</i>	статический метод	Сортировка элементов массива

Для перечисленных членов класса *Array* не указываются параметры, т.к. большинство из них имеют несколько перегрузок.

Вызов статических методов осуществляется через обращение к имени класса, с использованием оператора «точка».

Например: **Array.Sort(myMass); Array.BinarySearch(myMass)**, где *Array* – имя класса, *Sort* и *BinarySearch* – имена статических методов. При обращении к статическим методам класса *Array*, массив передается в метод как параметр. Например, при обращении к методам: **Array.Sort(myMass); Array.BinarySearch(myMass)**, массив **myMass** передается как параметр.

При обращении к свойствам или вызове экземплярного метода класса *Array*, осуществляется обращение к экземпляру класса, например, *myMass.Length*, или *myMass.SetValue()*.

Пример 13. Использование метода *BinarySearch()* - «Двоичный поиск в отсортированном массиве».

Дан массив pr:

Индекс	0	1	2	3	4
Элементы массива	3	8	12	30	84

Найти индекс элемента со значением 12 в отсортированном массиве pr:

```
int index = Array.BinarySearch(pr, 12);  
Console.WriteLine("index=" + index);
```

Результат работы метода **BinarySearch** – индекс элемента со значением 12, равный 2.

Пример 14. Использование экземплярного метода *SetValue()* – «Установка значения элемента массива».

Дан массив pr:

Индекс	0	1	2	3	4
Элементы массива	3	8	12	30	84

После применения экземплярного метода *pr.SetValue(45,0)*, где «45» – значение элемента массива, «0» - индекс элемента массива pr. Результат работы данного метода:

Индекс	0	1	2	3	4
Элементы массива	45	8	12	30	84

Задания для выполнения в аудитории

Задание №1. Разработайте консольное приложение для выполнения следующих действий над одномерным целочисленным массивом:

- *Расчет:*
 - суммы;
 - произведения;
 - среднего арифметического значения элементов массива
- *Вывод* на экран четных элементов массива.
- *Сортировка* элементов массива, используя *свойства и методы* класса Array.

Предусмотрите для массива:

- размерность – в константах;
- инициализация – «случайным образом»;
- вывод на экран исходного массива;
- вывод на экран отсортированного массива.

Наличие методов:

- заполнения элементов массива случайным образом;
- вывода массива на экран (метод с параметром, в качестве параметра используется массив).

Задания для самостоятельной работы студента (СРС)

Задание №1. Выполните задание согласно Вашему варианту

Вариант	Задание
1	<p>Разработать отдельный класс MASSIV_1 для работы с одномерным массивом, состоящим из N1 вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N1 – размерность массива; S1 – сумма; P1– произведение.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «размерность массива должна быть положительной». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА» .</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вычисление суммы отрицательных элементов массива; – вычисление произведения элементов массива, расположенных между максимальным и минимальным элементами массива.
2	<p>Разработать отдельный класс MASSIV_2 для работы с одномерным массивом, состоящим из N2 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N2 – размерность массива; S2 – сумма; P2 – произведение.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «количество элементов массива должно быть не менее 20». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА» .</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вычисление суммы положительных элементов массива; – вычисление произведения элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами массива.
3	<p>Разработать отдельный класс MASSIV_3 для работы с одномерным массивом, состоящим из N3 вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N3 – размерность массива; S3 – сумма; P3– произведение.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «количество элементов массива должно быть не более 100». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА» .</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вычисление суммы элементов массива, значения которых больше K (K введите с клавиатуры); – вычисление произведения элементов массива с четными индексами.

Вариант	Задание
4	<p>Разработать отдельный класс MASSIV_4 для работы с одномерным массивом, состоящим из N4 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N4 – размерность массива; SR4– среднее арифметическое; K4– количество положительных элементов массива.</p> <p><i>Свойство:</i> для получения количества положительных элементов массива.</p> <p>Для работы со свойствами рассмотрите Пример 1, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вычисление среднего арифметического элементов массива с нечетными индексами.
5	<p>Разработать отдельный класс MASSIV_5 для работы с одномерным массивом, состоящим из N5 вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N5 – размерность массива; S5 – сумма; P5– произведение.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «размерность массива должна быть положительной». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вычисление суммы модулей элементов массива, расположенных после первого отрицательного элемента; – вычисление произведения элементов массива, больших 5.
6	<p>Разработать отдельный класс MASSIV_6 для работы с одномерным массивом, состоящим из N6 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N6 – размерность массива; S6 – сумма.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «количество элементов массива должно быть равно 30». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вывода на экран четных элементов массива; – вычисление суммы элементов массива, значения которых меньше M1 и больше M2 (M1, M2 вводятся с клавиатуры);
7	<p>Разработать отдельный класс MASSIV_7 для работы с одномерным массивом, состоящим из N7 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N7 – размерность массива; S7– произведение; K7– количество элементов массива, модуль которых больше 17.</p> <p><i>Свойство:</i> для получения количества элементов массива, модуль которых больше 17.</p>

Вариант	Задание
	<p>Для работы со свойствами рассмотрите Пример 1, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вычисление суммы элементов массива, расположенных между максимальным и минимальным элементами массива.
8	<p>Разработать отдельный класс MASSIV_8 для работы с одномерным массивом, состоящим из N8 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N8 – размерность массива; S8 – сумма; K8– количество положительных элементов массива .</p> <p><i>Свойство:</i> для получения количества положительных элементов массива. Для работы со свойствами рассмотрите Пример 1, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); предусмотреть наличие в файле одного элемента, со значением ноль; – вывода массива на экран; – вычисление произведения элементов массива, расположенных до первого элементами массива, равного нулю.
9	<p>Разработать отдельный класс MASSIV_9 для работы с одномерным массивом, состоящим из N9 вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N9 – размерность массива; S9 – сумма.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «размерность массива должна быть положительной». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА» .</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); предусмотреть наличие в файле как положительных, так и отрицательных элементов; – вывода массива на экран; – вывода на экран индекс первого минимального по модулю элемента массива; – вычисление суммы элементов массива, значения которых кратны пяти;
10	<p>Разработать отдельный класс MASSIV_10 для работы с одномерным массивом, состоящим из N10 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N10 – размерность массива; P10– произведение; K10– количество отрицательных элементов массива .</p> <p><i>Свойство:</i> для получения количества отрицательных элементов массива. Для работы со свойствами рассмотрите Пример 1, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); предусмотреть наличие в файле одного элемента, со значением ноль; – вывода массива на экран; – вычисление произведения элементов массива, расположенных после первого элементами массива, равного нулю.

Вариант	Задание
11	<p>Разработать отдельный класс MASSIV_11 для работы с одномерным массивом, состоящим из N11 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N11 – размерность массива; SR11– среднее геометрическое; K4– количество элементов массива.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «количество элементов массива должно быть не более 11». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА» .</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); предусмотреть в файле элементы, значения каждого из которых не должно превышать число восемь; – вывода массива на экран; – вычисление среднего геометрического элементов массива: $a_{\text{ср.геометр.}} = \sqrt[n]{a_1 * a_2 * ... * a_n} .$
12	<p>Разработать отдельный класс MASSIV_12 для работы с одномерным массивом, состоящим из N12 целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N12 – размерность массива; P12– произведение; K12– количества ненулевых элементов массива .</p> <p><i>Свойство:</i> для получения количества ненулевых элементов массива. Для работы со свойствами рассмотрите Пример 1, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); предусмотреть наличие в файле одного элемента, со значением ноль; – вывода массива на экран; – вычисление произведения элементов массива, расположенных после первого максимального элемента массива.
13	<p>Разработать отдельный класс MASSIV_13 для работы с одномерным массивом, состоящим из N13 вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N13 – размерность массива; S13 – сумма; P13– произведение.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «размерность массива должна быть положительной». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА» .</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); – вывода массива на экран; – вычисление суммы нечетных элементов массива; – вычисление произведения элементов массива с четными индексами.

Вариант	Задание
14	<p>Разработать отдельный класс MASSIV_14 для работы с одномерным массивом, состоящим из N14 вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N14 – размерность массива; P14– произведение; K14– количества нулевых элементов массива.</p> <p><i>Свойство:</i> для получения количества нулевых элементов массива. Для работы со свойствами рассмотрите Пример 1, раздел «СВОЙСТВА».</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); предусмотреть наличие в файле несколько элементов со значением ноль; – вывода массива на экран; – вычисление произведения максимального и минимального элементов массива.
15	<p>Разработать отдельный класс MASSIV_15 для работы с одномерным массивом, состоящим из N15 вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N15 – размерность массива; S15 – сумма.</p> <p><i>Свойство:</i> для установки и получения значения размерности массива со следующим ограничением: «количество элементов массива должно быть не менее 15». Для работы со свойствами рассмотрите Пример 2, раздел «СВОЙСТВА» .</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнения элементов массива из файла (см. Пример 6); предусмотреть наличие в файле как положительных, так и отрицательных элементов; – вывода массива на экран; – вывода на экран индекса первого максимального по модулю элемента массива; – вычисление суммы элементов массива, индексы которых кратны трем.

ВСТАВКА И УДАЛЕНИЕ ЭЛЕМЕНТОВ МАССИВА

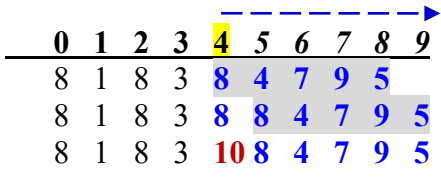
1. Вставка элемента в массив

Пример 1. Вставка числа 10, пятым элементом(индекс=4) массива.

Алгоритм:

- начиная с последнего элемента массива, по элемент с индексом 4 (до элемента с индексом 3), осуществляется сдвиг всех элементов *вправо* на одну позицию;
- замена значения элемента массива с индексом 4 на значение 10.

В данном примере: сдвиг всех элементов, начиная с индекса 8 по индекс 4.

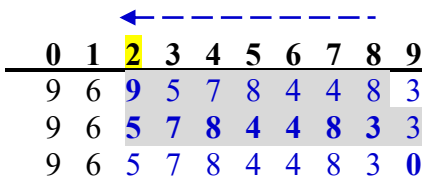
Массив	Фрагмент кода
	<pre>for (int i = 9; i > 4; i--) { pr[i] = pr[i-1]; //сдвиг элементов вправо } pr[4] = 10; //замена или: for (int i = 9; i > 4; i--) pr[i] = pr[i-1]; pr[4] = 10;</pre>

2. Удаление элемента массива

Пример 2. Удаление элемента массива с заданным номером (под номером 3, индекс 2).

Алгоритм:

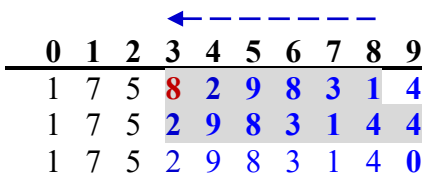
- сдвиг *влево* на одну позицию всех элементов, начиная с индекса 2 до предпоследнего элемента;
- замена последнего элемента массива (индекс 9) на ноль

Массив	Фрагмент кода
	<pre>for (int i = 2; i < 8; i++) { // сдвиг влево элементов, стоящих //справа от удаляемого элемента pr [i] = pr [i + 1]; } pr [9] = 0; //замена или: for (int i = 2; i < 8; i++) pr [i] = pr [i + 1]; pr [9] = 0;</pre>

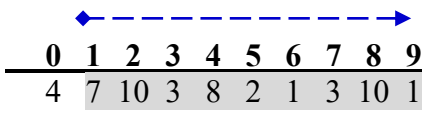
Пример 3. Удаление из массива элемента с заданным значением (элемент со значением = 8)

Алгоритм:

- поиск элемента со значением 8;
- сдвиг *влево* на одну позицию всех элементов, расположенных справа от удаляемого элемента (начиная с индекса 3 до предпоследнего элемента);
- замена последнего элемента массива (индекс 9) на ноль.

Массив	Фрагмент кода
 <p>0 1 2 3 4 5 6 7 8 9</p> <p>1 7 5 8 2 9 8 3 1 4</p> <p>1 7 5 2 9 8 3 1 4 4</p> <p>1 7 5 2 9 8 3 1 4 0</p>	<pre> int q = 8; for (int i = 0; i < pr.Length; i++) { // поиск if (q == pr [i]) { //сдвиг влево элементов, стоящих справа от //удаляемого элемента for (int k = i; k < pr.Length; k++) pr [k]= pr [k + 1]; pr [9] = 0; //замена } } </pre>

Пример 4. Циклический сдвиг всех элементов массива на одну позицию вправо

Массив	Фрагмент кода
 <p>0 1 2 3 4 5 6 7 8 9</p> <p>4 7 10 3 8 2 1 3 10 1</p> <p>Результат сдвига:</p> <p>0 1 2 3 4 5 6 7 8 9</p> <p>1 4 7 10 3 8 2 1 3 10</p>	<pre> int t = pr[9]; // сохранить последний элемент массива for (int i = 9; i >= 1; i--) pr [i] = pr [i - 1]; pr [0] = t; // первому элементу присваивовать //значение последнего элемента массива </pre>

Задания для выполнения в аудитории

Задание №1. Выполните задание согласно Вашему варианту

Вариант	Задание
1	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N1 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [10;20]; - вывод на экран элементов массива; - вставка значения минимального элемента перед элементом со значением K1; - вывод на экран элементов массива; - вставка в полученный массив, пять раз числа 100, после элемента со значением K1; - вывод на экран элементов массива. <p>Значения N1 и K1 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
2	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N2 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [-15;25]; - вывод на экран элементов массива; - вставка после каждого отрицательного элемента массива, элемент со значением K2; - вывод на экран элементов массива; - вставка в начало полученного массива, шесть раз числа 0; - вывод на экран элементов массива. <p>Значения N2 и K2 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
3	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N3 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [3;200]; - вывод на экран элементов массива; - вставка числа со значением K3 после всех элементов, кратных 3; - вывод на экран элементов массива; - вставка, в полученный массив семь раз числа « - 5» перед элементом со значением K3; - вывод на экран элементов массива. <p>Значения N3 и K3 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
4	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N4 элементов вещественного типа:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [-14,00; 28,00]; - вывод на экран элементов массива с двумя знаками после запятой (см. «Вывод элементов массива на экран» Пример 6); - вставка числа, равного сумме элементов массива, пять раз, в начало массива; - вывод на экран элементов массива. <p>Значение N12 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>

Вариант	Задание
5	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N5 элементов целого типа:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [5; 25]; - вывод на экран элементов массива; - вставка числа со значением K5 после всех четных элементов массива; - вывод на экран элементов массива; - вставка в конец полученного массива десять раз число «55»; - вывод на экран элементов массива. <p>Значение N5 введите с клавиатуры; значение K5 – генерируется с помощью генератора случайных чисел. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
6	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N6 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [-16; 32]; - вывод на экран элементов массива; - продублировать все отрицательные элементы массива; - вывод на экран элементов массива; - вставка, в конец полученного массива, восемь раз числа K6; - вывод на экран элементов массива. <p>Значения N6 и K6 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
7	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N7 элементов вещественного типа:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [-7,00; 170,00]; - вывод на экран элементов массива с двумя знаками после запятой (см. «Вывод элементов массива на экран» Пример 6); - вставка числа со значением K7, перед всеми положительными элементами массива; - вывод на экран элементов массива; - вставка, в конец полученного массива, семь раз число «0». - вывод на экран элементов массива. <p>Значение N7 введите с клавиатуры; значение K7 – генерируется с помощью генератора случайных чисел. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
8	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N8 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [8; 80]; - вывод на экран элементов массива; - вставка после максимального элемента массива число со значением K8; - вывод на экран элементов массива; - вставка в полученный массив десять раз числа «0» перед максимальным элементом массива; - вывод на экран элементов массива. <p>Значение N8 введите с клавиатуры; значение K8 – генерируется с помощью генератора случайных чисел в диапазоне [- 10; -5]. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>

Вариант	Задание
9	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N9 элементов целого типа:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [1; 9]; - вывод на экран элементов массива; - вставка числа, равного произведению элементов массива, в конец массива; - вывод на экран элементов массива; - вставка в полученный массив девять раз число K9 перед элементом массива с индексом 4. <p>Значения N9 и K9 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
10	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N10 элементов вещественного типа:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [2,00; 10,00]; - вывод на экран элементов массива с двумя знаками после запятой (см. «Вывод элементов массива на экран» Пример 6); - вставка числа со значением K10, перед всеми элементами массива, которые меньше 6; - вывод на экран элементов массива; - вставка, в конец полученного массива семь раз число « - 30». - вывод на экран элементов массива. <p>Значение N10 введите с клавиатуры; значение K10 – генерируется с помощью генератора случайных чисел. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
11	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N11 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [-11;110]; - вывод на экран элементов массива; - продублировать все элементы массива, кратные 4; - вывод на экран элементов массива; - вставка, в начало полученного массива девять раз число K11; - вывод на экран элементов массива. <p>Значения N11 и K11 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
12	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N12 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [0;120]; - вывод на экран элементов массива; - вставка значения максимального элемента после элемента со значением K12; - вывод на экран элементов массива; - вставка, в полученный массив восемь раз число « - 150», после элемента со значением K12; - вывод на экран элементов массива. <p>Значения N12 и K12 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>

Вариант	Задание
13	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N13 элементов целого типа:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [13; 33]; - вывод на экран элементов массива; - вставка числа, равного произведению четных элементов массива, в конец массива; - вывод на экран элементов массива; - вставка в полученный массив десять раз число K13 после элемента массива с индексом 5. <p>Значения N13 и K13 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
14	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N14 целочисленных элементов:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [-30;44]; - вывод на экран элементов массива; - вставка после каждого положительного элемента массива элемент со значением K14; - вывод на экран элементов массива; - вставка в конец полученного массива, семь раз число 14; - вывод на экран элементов массива. <p>Значения N14 и K14 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>
15	<p>Разработайте программу для выполнения следующих действий над элементами одномерного массива из N15 элементов вещественного типа:</p> <ul style="list-style-type: none"> - заполнение массива «случайным образом», в диапазоне [-15,00; 15,00]; - вывод на экран элементов массива с двумя знаками после запятой (см. «Вывод элементов массива на экран» Пример 6); - вставка числа, равного значению среднего арифметического элементов массива, восемь раз, в конец массива; - вывод на экран элементов массива. <p>Значения N15 введите с клавиатуры. Вывод на экран элементов массива, оформите как отдельный метод с параметром; в качестве параметра – массив.</p>

Задания для самостоятельной работы студента (СРС)

Задание №1. Разработайте программу для выполнения указанных действий над элементами одномерного массива, согласно Вашему варианту. Предусмотреть вывод на экран массива после каждого изменения массива.

Перед выполнением задания повторите раздел «НЕСТАТИЧЕСКИЕ МЕТОДЫ», Пример 3; модификаторы доступа.

Вариант	Задание
1	<p>Разработать отдельный класс MASSIV_1 для работы с одномерным массивом вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N1 – размерность массива;</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотреть наличие в файле как положительных, так и отрицательных элементов – вывод массива на экран; – удаление первого отрицательного элемента массива; – удаление последних пяти элементов массива; – реализации циклического сдвига элементов массива вправо на 9 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно.</p> <p>Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен.</p> <p>Класс Program обязательно должен содержать объявление (описание) массива.</p>
2	<p>Разработать отдельный класс MASSIV_2 для работы с одномерным массивом вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N2 – размерность массива;</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); – вывод массива на экран; – удаление всех элементов массива с четными индексами; – удаление элемента массива со значением K2, введенного с клавиатуры; – реализация циклического сдвига элементов массива влево на 5 элементов; – записи измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно.</p> <p>Модификаторы доступа для методов должны обеспечить доступ ко всем методам из любого метода любого класса программы.</p> <p>Класс Program обязательно должен содержать объявление (описание) массива.</p>
3	<p>Разработать отдельный класс MASSIV_3 для работы с одномерным массивом вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N3 – размерность массива;</p> <p><i>Методы:</i></p>

Вариант	Задание
	<ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотреть наличие в файле как положительных, так и отрицательных элементов – вывод массива на экран; – удаление первого отрицательного элемента массива; – удаление элементов массива, индексы которых находятся в диапазоне [3;13]; – реализация циклического сдвига элементов массива вправо на 8 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен. Класс Program обязательно должен содержать объявление (описание) массива.</p>
4	<p>Разработать отдельный класс MASSIV_4 для работы с одномерным массивом целочисленных элементов. <i>Элементы класса:</i> <i>Поля:</i> N4 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов; – вывод массива на экран; – удаление всех элементов массива, расположенных после элемента со значением K4, введенного с клавиатуры; – удаление элемента массива со значением 44; – реализация циклического сдвига элементов массива влево на 6 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам из любого метода любого класса программы. Класс Program обязательно должен содержать объявление (описание) массива.</p>
5	<p>Разработать отдельный класс MASSIV_5 для работы с одномерным массивом вещественных элементов. <i>Элементы класса:</i> <i>Поля:</i> N5 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотреть наличие в файле как положительных, так и отрицательных элементов – вывод массива на экран; – удаление последнего положительного элемента массива; – удаление первых восьми элементов массива; – реализация циклического сдвига элементов массива вправо на 5 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно.</p>

Вариант	Задание
	<p>Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен.</p> <p>Класс Program обязательно должен содержать объявление (описание) массива.</p>
6	<p>Разработать отдельный класс MASSIV_6 для работы с одномерным массивом вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N6 – размерность массива;</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотреть наличие в файле как положительных, так и отрицательных элементов; – вывод массива на экран; – удаление всех элементов массива с нечетными индексами; – удаление последнего отрицательного элемента массива; – реализация циклического сдвига элементов массива влево на 6 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно.</p> <p>Модификаторы доступа для методов должны обеспечить доступ ко всем методам, из любого метода любого класса программы.</p> <p>Класс Program обязательно должен содержать объявление (описание) массива.</p>
7	<p>Разработать отдельный класс MASSIV_7 для работы с одномерным массивом целочисленных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N7 – размерность массива;</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов; – вывод массива на экран; – удаление всех элементов массива, индексы которых принадлежат диапазону [7, 17]; – удаление минимального элемента массива; – реализация циклического сдвига элементов массива вправо на 7 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно.</p> <p>Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен.</p> <p>Класс Program обязательно должен содержать объявление (описание) массива.</p>
8	<p>Разработать отдельный класс MASSIV_8 для работы с одномерным массивом вещественных элементов.</p> <p><i>Элементы класса:</i></p> <p><i>Поля:</i> N8 – размерность массива;</p> <p><i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для

Вариант	Задание
	<p>реализации методов;</p> <ul style="list-style-type: none"> – вывод массива на экран; – удаление всех отрицательных элементов массива, с индексами большими 8; – удаление всех элементов массива больших 18; – реализация циклического сдвига элементов массива влево на 8 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам, из любого метода любого класса программы. Класс Program обязательно должен содержать объявление (описание) массива.</p>
9	<p>Разработать отдельный класс MASSIV_9 для работы с одномерным массивом целочисленных элементов.</p> <p><i>Элементы класса:</i> <i>Поля:</i> N9 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов; – вывод массива на экран; – удаление всех элементов массива, принадлежащих диапазону [10, 39]; – удаление элемента массива со значением K9, введенного с клавиатуры; – реализация циклического сдвига элементов массива вправо на 6 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен. Класс Program обязательно должен содержать объявление (описание) массива.</p>
10	<p>Разработать отдельный класс MASSIV_10 для работы с одномерным массивом целочисленных элементов.</p> <p><i>Элементы класса:</i> <i>Поля:</i> N10 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов; – вывод массива на экран; – удаление всех элементов массива, индексы которых принадлежат диапазону [10, 15]; – удаление максимального элемента массива; – реализация циклического сдвига элементов массива влево на 5 элементов; – запись измененного массива в новый файл (см. Приложение 2) <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов, должны обеспечить доступ ко всем</p>

Вариант	Задание
	<p>методам, из любого метода любого класса программы. Класс Program обязательно должен содержать объявление (описание) массива.</p>
11	<p>Разработать отдельный класс MASSIV_11 для работы с одномерным массивом вещественных элементов. <i>Элементы класса:</i> <i>Поля:</i> N11 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов; – вывод массива на экран; – удаление элемента массива с индексом K11, введенным с клавиатуры; – удаление первых одиннадцати элементов массива; – реализация циклического сдвига элементов массива вправо на 9 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен. Класс Program обязательно должен содержать объявление (описание) массива.</p>
12	<p>Разработать отдельный класс MASSIV_12 для работы с одномерным массивом вещественных элементов. <i>Элементы класса:</i> <i>Поля:</i> N12 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов; – вывод массива на экран; – удаление всех элементов массива, с индексами, кратными пяти; – удаление всех элементов массива меньших 120; – реализация циклического сдвига элементов массива влево на 7 элементов; – записи измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам, из любого метода любого класса программы. Класс Program обязательно должен содержать объявление (описание) массива.</p>
13	<p>Разработать отдельный класс MASSIV_13 для работы с одномерным массивом вещественных элементов. <i>Элементы класса:</i> <i>Поля:</i> N13 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов;

Вариант	Задание
	<ul style="list-style-type: none"> – вывод массива на экран; – удаление всех положительных элементов массива; – удаление последних девяти элементов массива; – реализация циклического сдвига элементов массива вправо на 8 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен. Класс Program обязательно должен содержать объявление (описание) массива.</p>
14	<p>Разработать отдельный класс MASSIV_14 для работы с одномерным массивом вещественных элементов. <i>Элементы класса:</i> <i>Поля:</i> N14 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); предусмотрите наличие в файле значений элементов массива, необходимых для реализации методов; – вывод массива на экран; – удаление всех элементов массива, индексы которых принадлежат диапазону [5, 14]; – удаление всех элементов массива больших 24; – реализация циклического сдвига элементов массива влево на 9 элементов; – записи измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам из любого метода любого класса программы. Класс Program обязательно должен содержать объявление (описание) массива.</p>
15	<p>Разработать отдельный класс MASSIV_15 для работы с одномерным массивом целочисленных элементов. <i>Элементы класса:</i> <i>Поля:</i> N15 – размерность массива; <i>Методы:</i></p> <ul style="list-style-type: none"> – заполнение элементов массива из файла (см. Пример 6); – вывод массива на экран; – удаление всех нечетных элементов массива; – удаление элемента массива с индексом K15, введенным с клавиатуры; – реализация циклического сдвига элементов массива вправо на 6 элементов; – запись измененного массива в новый файл (см. Приложение 2). <p>Модификаторы доступа для полей определите самостоятельно. Модификаторы доступа для методов должны обеспечить доступ ко всем методам только внутри сборки, в которой этот метод определен. Класс Program обязательно должен содержать объявление (описание) массива.</p>

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
2. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM
3. Чтение и запись в текстовый файл с помощью Visual C#. URL: <https://docs.microsoft.com/ru-ru/troubleshoot/dotnet/csharp/read-write-text-file>
4. С# и NET Чтение и запись текстовых файлов. URL: <https://metanit.com/sharp/tutorial/5.5.php>

Приложение 1

Некоторые методы класса Random

Метод	Способ вызова	Описание
Next	Next()	Возвращает положительное целое число из всего диапазона int
Next	Next(int maxValue)	Возвращает положительное целое число из диапазона [0, maxValue - 1]
Next	Next(int minValue, int maxValue)	Возвращает положительное целое число из диапазона [minValue, maxValue]
NextDouble	NextDouble()	Возвращает вещественное число из диапазона [0,1)
NextBytes	NextBytes(byte[] buffer)	Позволяет генерировать последовательность случайных чисел из всего диапазона byte, которую записывает в одномерный массив buffer

Приложение 2

Запись элементов массива в текстовый файл

Файл не создан заранее на диске. Файл создается "в тексте программы".

Дано:

- N – количество элементов массива pr;
- Исходный массив вещественных элементов, заполнен следующим образом:

```
for (int i = 0; i < N; i++) pr[i] = i+20.15;
```

Требуется: Записать элементы массива pr в текстовый файл.

Решение:

1. Создать объект, например, f1 класса StreamWriter пространства имен System.IO :

`StreamWriter f1 = new StreamWriter("3.txt");` - файл создается для записи, в текущей папке(папка проекта)

`StreamWriter f1 = new StreamWriter("D:/3.txt");` - файл создается для записи, на диске D.

2. Запись в файл элемента массива pr, с использованием объекта f1:

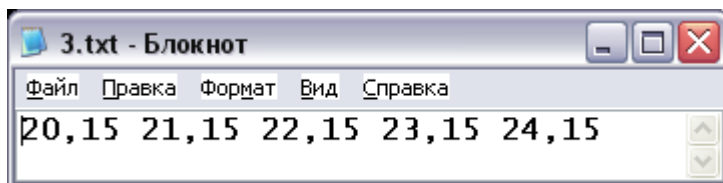
`f1.Write(pr[0]);` //запись, например, нулевого элемента массива в файл 3.txt

Фрагмент кода:

```
StreamWriter f1 = new StreamWriter("D:/3.txt");  
for (int i = 0; i < .N; i = i + 1) f1.Write(pr[i]+" "); //запись, через пробел, i-того элемента  
//массива в файл 3.txt  
f1.Close(); // завершить работу с файлом 3.txt
```

Результат:

текстовый файл с записанными в него элементами, вещественного типа, массива pr



СОРТИРОВКИ ПРОСТЫЕ

- 1) $A[1] < A[2] < \dots < A[N]$ – сортировка по возрастанию
- 2) $A[1] \leq A[2] \leq \dots \leq A[N]$ – сортировка по «неубыванию»
- 3) $A[1] > A[2] > \dots > A[N]$ – сортировка по убыванию
- 4) $A[1] \geq A[2] \geq \dots \geq A[N]$ – сортировка по «невозрастанию»

Все алгоритмы сортировок описывают сортировку по возрастанию.

1. «Простые вставки»

Алгоритм:

1. На j -м шаге считается, что часть массива, содержащая первые $j-1$ элементов, уже упорядочена, т.е. $A[1] \leq A[2] \leq \dots \leq A[j-1]$.
2. Берется j -й элемент из не отсортированной части массива, и для него подбирается место в отсортированной части массива такое, что после его вставки упорядоченность не нарушается, т.е. требуется найти такое $j(0 \leq j \leq (j-1))$, что $A[j] \leq A[j] \leq A[j+1]$.
3. Выполняется вставка элемента $A[j]$ массива на место j .

Примечание: на каждом шаге отсортированная часть массива увеличивается. Для выполнения полной сортировки потребуется выполнить $N-1$ шаг.

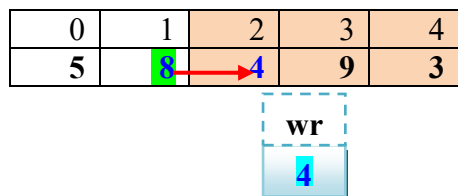
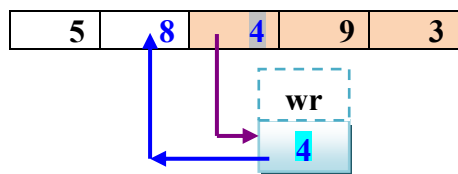
Фрагмент кода:

```
for (int j = 1; j < N; j++)
{
    i = j-1;
    wr = A[j];
    while ((i>=0)&&(A[i]>wr))
    {
        A[i+1] = A[i];
        i = i - 1;
    }
    A[i+1] = wr;
}
```

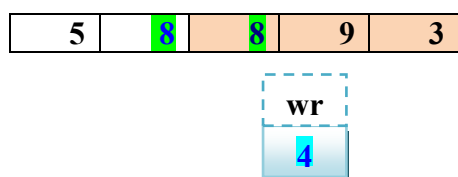
Описание алгоритма на примере массива из пяти элементов;

1. Работа цикла for, для j=1																								
<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;"> <p>Отсортированная часть массива</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>5</td><td>8</td><td>4</td><td>9</td><td>3</td></tr> </table> <p>wr</p> <p>8</p> </div> <div style="text-align: center;"> <p>Отсортированная часть массива</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>5</td><td>8</td><td>4</td><td>9</td><td>3</td></tr> </table> </div> </div>					0	1	2	3	4	5	8	4	9	3	0	1	2	3	4	5	8	4	9	3
0	1	2	3	4																				
5	8	4	9	3																				
0	1	2	3	4																				
5	8	4	9	3																				

2. Работа цикла for, для j=2



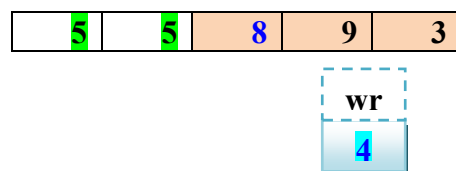
Массив после сдвига элемента со значением 8:



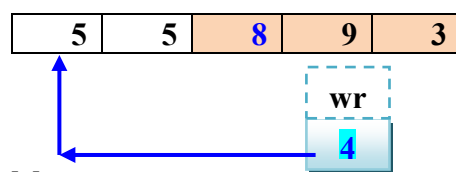
0	1	2	3	4
5	8	8	9	3



Массив после сдвига элемента со значением 5:



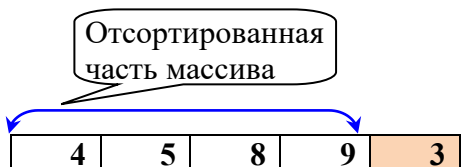
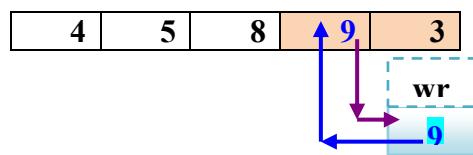
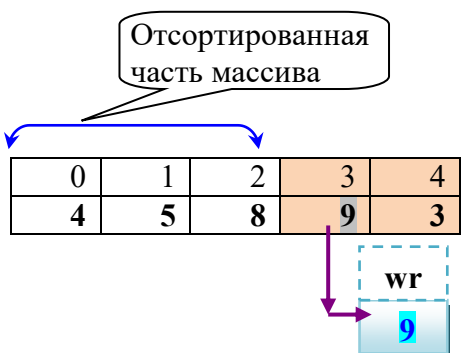
Вставка 4 вместо первого элемента (вместо 5):



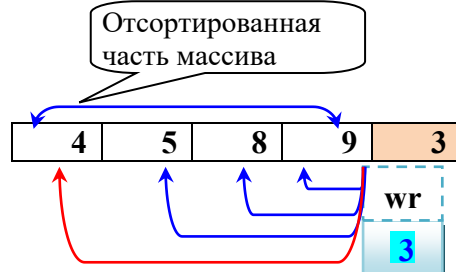
Массив после вставки элемента 4:

4	5	8	9	3
---	---	---	---	---

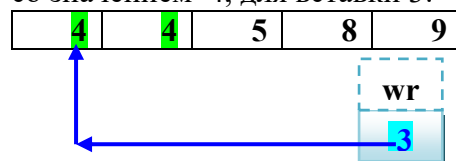
3. Работа цикла for, для j=3



4. Работа цикла for, для j=4



Массив после сдвига элемента со значением 4, для вставки 3:



Массив после вставки элемента со значением 3:

3	4	5	8	9
---	---	---	---	---

Отсортированный массив:

3	4	5	8	9
---	---	---	---	---

Трассировка алгоритма «простыми вставками», в соответствии с программным кодом см. Приложение 1.

2. «Простой выбор»

Алгоритм:

1. Осуществляется поиск *минимального* элемента среди всех еще неупорядоченных элементов массива.
2. Обмен местами: найденного минимального элемента и первого "поочередно" не отсортированным элементом.
3. Проводится сортировка оставшейся части массива, исключив из рассмотрения уже отсортированные элементы. К концу работы этого алгоритма последний (N-й) элемент массива автоматически окажется максимальным.

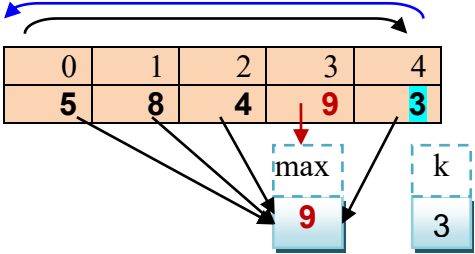
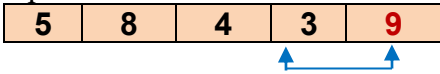
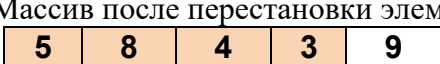
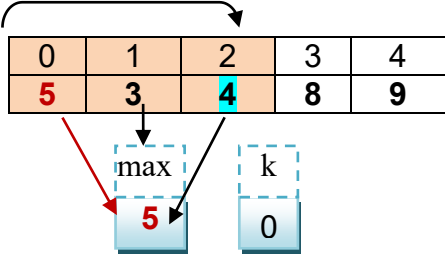
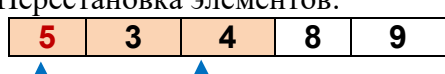
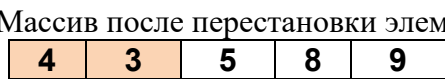
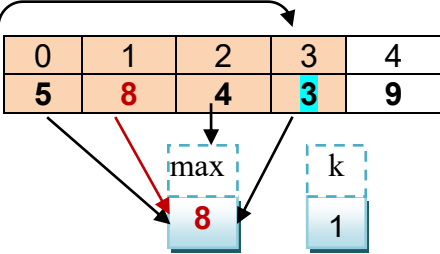
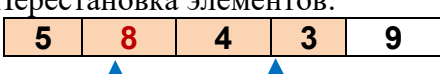
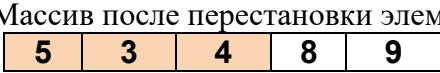
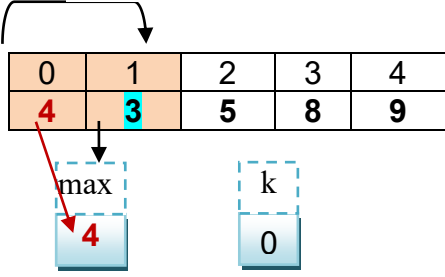
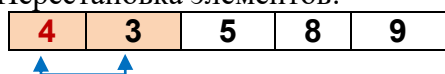

Фрагмент кода:

N – количество элементов массива;

max – значение максимального элемента, рассматриваемой части массива

```
for (j = N - 1; j >= 1; j=j - 1) //Просмотр массива для поиска максимума
{
    int max = A[j]; //начальное значение максимума в рассматриваемой части массива
    int k = j;      // начальное значение номера максимума в рассматриваемой части
                    //массива
    for (i = 0; i < j; i=i+1) // поиск элемента большего чем максимум
    {
        if (A[i] > max)
        {
            max = A[i]; // найден больший элемент чем максимум
            k = i;      // фиксируется номер максимума
        }
    }
    // перестановка элементов
    A[k] = A[j];
    A[j] = max;
}
```

Описание алгоритма на примере массива из пяти элементов:

<p>1. Работа цикла for, для j=4</p> <p>Поиск максимального значения j=4...0; i=0...4</p>  <p>Перестановка элементов:</p>  <p>Массив после перестановки элементов:</p> 	<p>3. Работа цикла for, для j=2</p> <p>Поиск максимального значения в неотсортированной части массива: i=0...2</p>  <p>Перестановка элементов:</p>  <p>Массив после перестановки элементов:</p> 
<p>2. Работа цикла for, для j=3</p> <p>Поиск максимального значения в неотсортированной части массива: i=0...3</p>  <p>Перестановка элементов:</p>  <p>Массив после перестановки элементов:</p> 	<p>4. Работа цикла for, для j=1</p> <p>Поиск максимального значения в неотсортированной части массива: i=0...1</p>  <p>Перестановка элементов:</p>  <p>Массив после перестановки элементов:</p>  <p>При j=0 внешний цикл for закончит работу. Массив отсортирован.</p>

Трассировка алгоритма «простым выбором», в соответствии с программным кодом см. Приложение 2.

Задания для выполнения в аудитории

Задание №1. Дан целочисленный массив из N элементов, заполненный «случайным образом». Используя алгоритм сортировки массива *«простыми вставками»*, разработайте программу для выполнения следующих действий над элементами массива:

1. Сортировка элементов массива *по возрастанию*, в виде отдельного метода SORT. (Вызов метода осуществляется из метода Main). В методе сортировки обеспечьте:
 - а) подсчет количества *сравнений* (KS) и подсчет количества *пересылок* (KP);
 - б) вывод KS и KP в методе сортировки.
2. Просмотр изменений в массиве после каждого шага сортировки.
3. Сортировка *по убыванию* элементов массива, с *четными* индексами, в виде отдельного метода SORT_1. (Вызов метода осуществляется из метода Main).

Задания для самостоятельной работы студента (СРС)

Задание №1. Дан целочисленный массив из N элементов. Используя алгоритм сортировки массива «*простым выбором*», разработайте программу, удовлетворяющую следующим требованиям:

1. Реализована сортировка элементов массива *по убыванию*, в виде отдельного *метода*; в метод массив должен передаваться из метода Main; после сортировки - «возвращаться» в метод Main.
2. Вывод отсортированного массива должен осуществляться в методе Main.
3. В методе сортировки обеспечены:
 - a) подсчет количества сравнений (KS) и подсчет количества перестановок (KP);
 - b) вывод KS и KP в методе сортировки;
 - c) просмотр изменений в массиве после каждого шага сортировки.
4. Предусмотрены методы заполнения массива: «худшим», «лучшим», «средним» способами.
5. Протестирована разработанная программа для N=10, при трех способах заполнения.

Задание №2

2.1. Рассмотрите представленный ниже материал:

Метод «пузырька» – метод простого обмена.

Производится последовательное упорядочение смежных пар элементов массива: X[1] и X[2], X[2] и X[3], ...X[N-1] и X[N]. Если в паре первый элемент больше второго, то элементы меняются местами. Далее, таким же образом, обрабатываем следующую пару. В итоге, после N-1 сравнения максимальное значение переместится на место элемента X[N], т.е. "вверх" окажется самый "легкий" элемент – отсюда аналогия с пузырьком. Следующий проход делается аналогичным образом до второго сверху элемента (X[N-1]), в результате второй по величине элемент поднимется на правильную позицию и т.д. Для сортировки всего массива нужно выполнить N-1 проход по массиву. При первом прохождении нужно сравнить N-1 пар элементов, при втором прохождении N-2 пары, при k-м прохождении (N - k) пар.

Фрагмент кода	Работа алгоритма на примере массива из 5 чисел
<p>С.Окулов «Основы программирования»:</p> <pre> for (i = 1; i < N; i++) for (j = 0; j < N - i; j++) if (X[j]>X[j+1]) { temp=x[j]; X[j]=X[j+1]; X[j+1]= temp; }; </pre>	<p>Массив: 6 8 0 4 4</p> <p>Сортировка: Просмотр 1:</p> <pre> 6 8 0 4 4 сравнение(обмена нет) 6 8 0 4 4 сравнение, обмен 6 0 8 4 4 сравнение, обмен 6 0 4 8 4 сравнение, обмен 6 0 4 4 8 результат обмена 6 0 4 4 8 </pre> <p>Просмотр 1 массива закончен</p>

Фрагмент кода	Работа алгоритма на примере массива из 5 чисел
В подобных реализациях внешний цикл выполняется N-1 раз, и не фиксирует, произошел обмен элементов массива или нет.	<p><i>Просмотр 2:</i></p> <pre> 6 0 4 4 8 сравнение, обмен 0 6 4 4 8 сравнение, обмен 0 4 6 4 8 сравнение, обмен 0 4 4 6 8 результат обмена 0 4 4 6 8 </pre> <p>Просмотр 2 массива закончен</p> <p><i>Просмотр 3:</i></p> <pre> 0 4 4 6 8 сравнение(обмена нет) 0 4 4 6 8 сравнение(обмена нет) </pre> <p>Просмотр 3 массива закончен</p> <p><i>Просмотр 4:</i></p> <pre> 0 4 4 6 8 сравнение(обмена нет) </pre> <p>Просмотр 4 массива закончен</p>

Д.Кнут в своей знаменитой книге «Искусство программирования», т.3, при рассмотрении алгоритма, вводит переменную t для фиксации обмена ($t=0$, если обмена нет).

Введение такой переменной позволяет уменьшить число «лишних» проходов по массиву.

Перед каждым проходом по внутреннему циклу, значение переменной t устанавливается, равным нулю, а после произошедшего обмена устанавливается равным 1. После окончания работы внутреннего цикла, переменная t сравнивается с 0: если значение t равно нулю, то обменов не было, следовательно, массив отсортирован, и можно выйти из внешнего цикла (досрочно выйти из внешнего цикла, т.е. завершить алгоритм сортировки).

2.2. Реализуйте сортировку обменом, предложенную Д.Кнутом, с учетом переменной t , фиксирующей состоявшиеся обмены элементов массива. Предусмотрите подсчет количества сравнений и перестановок элементов массива.

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

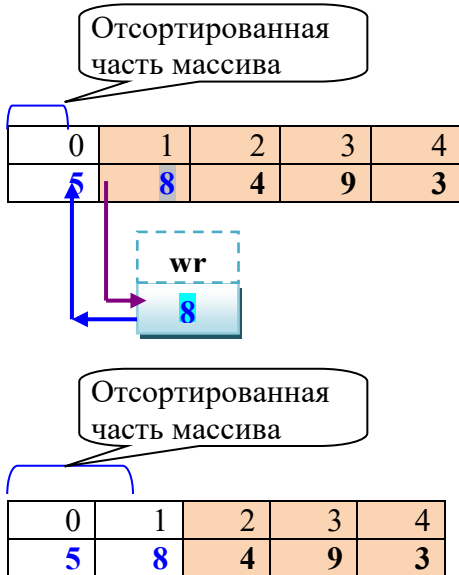
1. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). 2-е изд. Москва: Вильямс, 2007. Т. 3. 832 с
2. Левитин А. В. Алгоритмы: введение в разработку и анализ. ; Пер. с англ. М. : Издательский дом Вильямс, 2006
3. Павловская Т.А. С#. Программирование на языке высокого уровня: учебник для вузов. СПб: Питер, 2009.
4. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM

Приложение 1

Сортировка простыми вставками

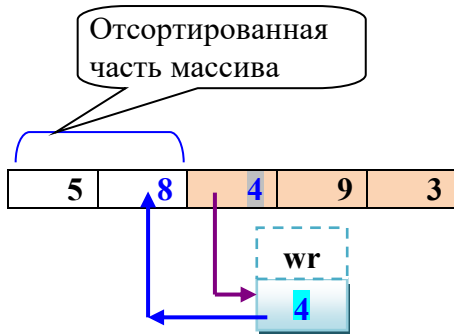
Описание алгоритма на примере массива из пяти элементов;

Работа цикла for, для j=1



```
for (int j = 1; 1 < N; j++) // неотсортированная часть
{
    i = 1-1; // i=0
    wr = A[1]; // wr=8
    while ((0>=0)&&(5>8))
    {
        A[i+1] = A[i];
        i = i - 1;
    }
    A[1] = wr; // A[1]=8
}
```

2. Работа цикла for, для j=2



```
for (int j = 2; 2 < N; j++)
{
    i = 2-1; // i=1
    wr = A[2]; // wr=4
    While ((1>=0)&&(A[1]>wr) // (1>=0)&&(8>4)
    {
        A[2] = A[1];
        i = 1 - 1;
    }
    A[1] = wr;
}
```



```
for (int j = 2; 2 < N; j++)
{
    i = 2-1; // i=1
    wr = A[2]; // wr=4
    //Поиск места для вставки числа 4 в
    отсортированную //часть массива, просматривая под-
    массив справа налево
    While ((1>=0)&&(A[1]>wr) // (1>=0)&&(8>4)
    { //8>4, тогда подготовка для вставки 4 вместо 8:
        A[2] = A[1]; //сдвиг вправо элемента 8
        i = 1 - 1; // i=0
    }
    A[1] = wr;
}
```

0	1	2	3	4
5	8	8	9	3



5	8	8	9	3
---	---	---	---	---



Массив после сдвига элемента со значением 5:

5	5	8	9	3
---	---	---	---	---



```
for (int j = 2; 2 < N; j++)
```

```
{
    i = 2-1;    // i=1
    wr = A[2];  // wr=4
```

//Поиск места для вставки числа 4 в отсортированную часть массива, просматривая подмассив справа налево

```
While ((i>=0)&&(A[i]>wr) //((i>=0)&&(5>4))
```

```
{ //5>4, тогда подготовка для вставки 4
```

```
    //вместо 5:
```

```
    A[1] = A[0]; //сдвиг вправо элемента 5
```

```
    i = 0 - 1;    // i= -1
```

```
}
```

```
A[1] = wr;
```

```
}
```

Вставка 4 вместо первого элемента (вместо 5):

5	5	8	9	3
---	---	---	---	---



Массив после вставки элемента 4:

4	5	8	9	3
---	---	---	---	---

```
for (int j = 2; 2 < N; j++)
```

```
{
    i = 2-1;    // i=1
```

```
    wr = A[2];  // wr=4
```

//Поиск места для вставки числа 4 в отсортированную часть массива, просматривая подмассив справа налево

```
While ((-1>=0)&&(A[-1]>wr)
```

```
{
```

```
    A[1] = A[0];
```

```
    i = 0 - 1;
```

```
}
```

```
    A[-1+1] = wr; //A[0]=4
```

```
}
```

3. Работа цикла for, для j=3

Отсортированная часть массива

0	1	2	3	4
4	5	8	9	3

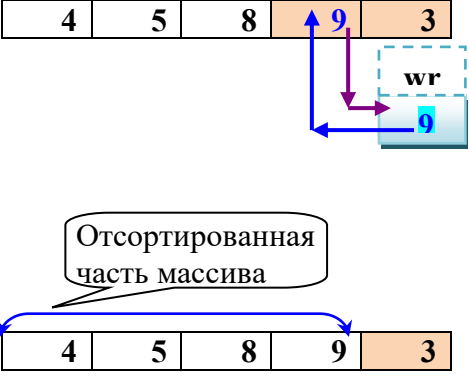
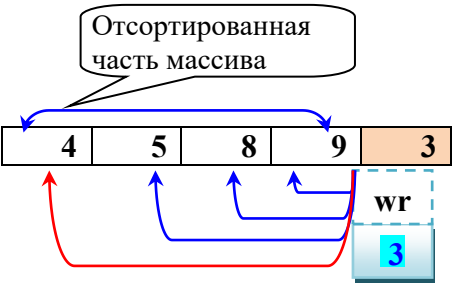
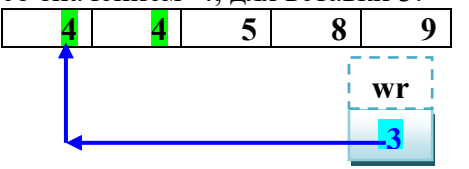
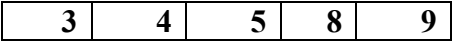
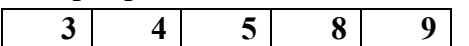


```
for (int j=3 ; 3< N;j++ )
```

```
{//Поиск места для вставки числа 9 в отсортированную часть массива
```

```
...
```

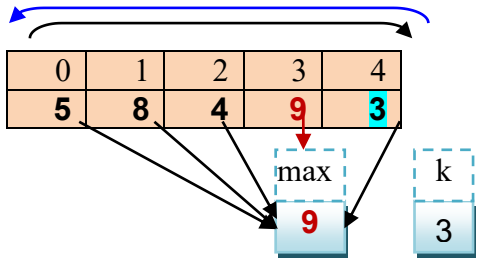
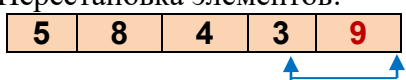
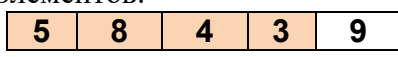
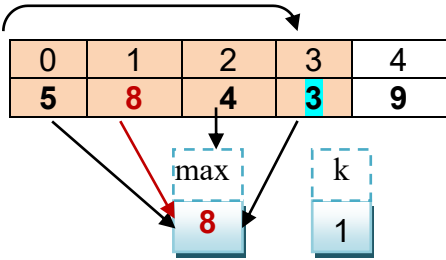
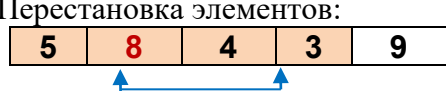
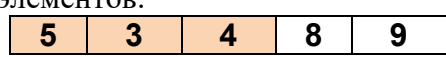
```
}
```

 <p>Отсортированная часть массива</p>	<p>Попытка вставить 9 вместо 8. Но $8 < 9$. Согласно алгоритму, происходит выполнение записи в $A[3]$ значения из переменной wr. Значение 9 находится «на своем месте»</p>
4. Работа цикла for, для $j=4$	
 <p>Отсортированная часть массива</p>	<pre>for (int j=4 ; 4< N; j++) { //Поиск места для вставки числа 3 в отсортированную часть массива ... }</pre> <p>Попытка вставить 3 : вместо 9; вместо 8; вместо 5; вместо 4</p>
<p>Массив после сдвига элемента со значением 4, для вставки 3:</p> 	
<p>Массив после вставки элемента со значением 3:</p>  <p>Отсортированный массив:</p> 	

Приложение 2

Сортировка простым выбором

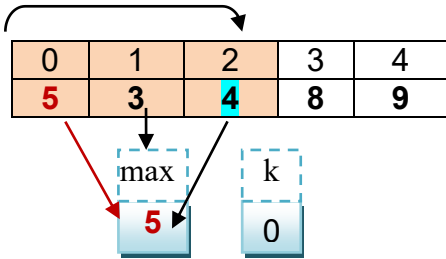
Описание алгоритма на примере массива из пяти элементов:

1. Работа цикла for, для j=4					
<p>Поиск максимального значения</p> <p>$j=4 \dots 0; \quad i=0 \dots 4$</p>  <p>Перестановка элементов:</p>  <p>Массив после перестановки элементов:</p> 	<pre> N=5 for (j = N-1; j >= 1; j=j-1) { int max = A[j]; int k = j; for (i = 0; i < j; i=i+1) { if (A[i] > max) { k = i; max = A[i]; } } A[k] = A[j]; A[j] = max; } </pre>	<p>j=4</p> <p>max = 3 k = 4 i=0; i < 4</p> <p>5 > 3</p> <p>k = 0 max = 5</p>	<p>i=1</p> <p>8 > 5</p> <p>k = 1 max = 8</p>	<p>i=2</p> <p>4 > 8</p>	<p>i=3</p> <p>9 > 8</p> <p>k = 3 max = 9</p> <p>i=4</p> <p>A[3] = A[4] A[4] = 9</p>
2. Работа цикла for, для j=3					
<p>Поиск максимального значения в неотсортированной части массива:</p> <p>$i=0 \dots 3$</p>  <p>Перестановка элементов:</p>  <p>Массив после перестановки элементов:</p> 	<pre> N=5 for (j = N-1; j >= 1; j=j-1) { int max = A[j]; int k = j; for (i = 0; i < j; i=i+1) { if (A[i] > max) { k = i; max = A[i]; } } A[k] = A[j]; A[j] = max; } </pre>	<p>j=3</p> <p>max = 3 k = 3 i=0; i < 3</p> <p>5 > 3</p> <p>k = 0 max = 5</p>	<p>i=1</p> <p>8 > 5</p> <p>k = 1 max = 8</p>	<p>i=2</p> <p>4 > 8</p>	<p>i=3</p> <p>A[1] = A[3] A[3] = 8</p>

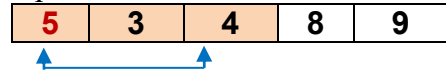
3. Работа цикла for, для j=2

Поиск максимального значения в неотсортированной части массива:

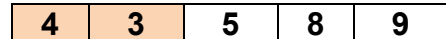
i=0...2



Перестановка элементов:



Массив после перестановки элементов:



```
N=5
for (j = N-1; j >=1; j=j-1)
{
    int max = A[j];
    int k = j;
    for (i = 0; i < j; i=i+1)
    {
        if (A[i] > max)
        {
            k = i;
            max = A[i];
        }
    }
    A[k] = A[j];
    A[j] = max;
}
```

j=2

max =4
k=2
i=0; i<2
5>4
k=0
max=5

i=1
3>5

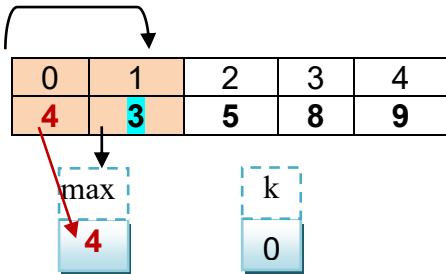
i=2

A[0]= A[2]
A[2]=5

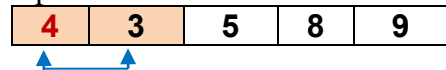
4. Работа цикла for, для j=1

Поиск максимального значения в неотсортированной части массива:

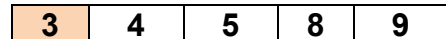
i=0...1



Перестановка элементов:



Массив после перестановки элементов:



```
N=5
for (j = N-1; j >=1; j=j-1)
{
    int max = A[j];
    int k = j;
    for (i = 0; i < j; i=i+1)
    {
        if (A[i] > max)
        {
            k = i;
            max = A[i];
        }
    }
    A[k] = A[j];
    A[j] = max;
}
```

j=1

max =3
k=1
i=0; i<1
4>3
k=0
max=4

i=1

A[0]= A[1]
A[1]=4

При j=0 внешний цикл for закончит работу. Массив отсортирован.

СОРТИРОВКИ БЫСТРЫЕ

СОРТИРОВКА СЛИЯНИЕМ

«Простое слияние»

Алгоритм слиянием был предложен Джоном фон Нейманом в 1945 г. Он является одним из самых быстрых способов сортировки.

Под *Слиянием* понимается объединение двух или более упорядоченных массивов в один упорядоченный. Особенностью этого алгоритма является то, что он работает с элементами массива преимущественно последовательно, благодаря чему именно этот алгоритм используется, например, при сортировке данных на жестком диске.

Данный алгоритм используется в тех случаях, когда для хранения промежуточных результатов есть возможность использовать память, сравнимую с размером исходного массива. Процесс слияния требует два отсортированных массива. Массив из одного элемента по определению является отсортированным.

Алгоритм реализации сортировки «Простым слиянием»:

1. *Разбиение.* Сортируемый массив *рекурсивно* разбивается на две части (два подмассива) примерно одинакового размера, до тех пор, пока в каждом из подмассивов останется по одному элементу. (*Разбиение*). Массив из одного элемента по определению является отсортированным.
2. *Слияние.* Две упорядоченные части (подмассивы) массива половинного размера соединяются в один. Вначале объединяются подмассивы, состоящие из одного элемента в каждом, в один массив из двух элементов, упорядоченных согласно условию сортировки. Отсортированные двухэлементные массивы *сливаются* в четырехэлементные массивы и так далее. Объединение фрагментов происходит, пока не образуется один упорядоченный массив (*Слияние частей*).

Пример работы алгоритма для массива из 8 элементов:

Индекс	0	1	2	3	4	5	6	7
Элемент массива	3	11	5	5	7	1	20	12

	Шаги алгоритма	Описание																
		<i>Разбиение</i>																
1	Массив А: (0+7)/2= 3 <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>11</td><td>5</td><td>5</td><td>7</td><td>1</td><td>20</td><td>12</td></tr></table>	0	1	2	3	4	5	6	7	3	11	5	5	7	1	20	12	Деление массива на левый подмассив (элементы с индексами с 0 по 3) и правый подмассив (элементы с индексами с 4 по 7)
0	1	2	3	4	5	6	7											
3	11	5	5	7	1	20	12											
2	(0+3)/2= 1 <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>11</td><td>5</td><td>5</td><td>7</td><td>1</td><td>20</td><td>12</td></tr></table>	0	1	2	3	4	5	6	7	3	11	5	5	7	1	20	12	Деление массива (3, 11, 5,5) на левый подмассив (3, 11) и правый подмассив (5, 5)
0	1	2	3	4	5	6	7											
3	11	5	5	7	1	20	12											

	Шаги алгоритма	Описание																
3	$(0+1)/2=0$ левый: <table><tr><td>0</td></tr><tr><td>3</td></tr></table> правый: <table><tr><td>1</td></tr><tr><td>11</td></tr></table>	0	3	1	11	Деление массива (3, 11) на левый (3) и правый (11). Границы начала и конца каждого подмассив совпадают. В каждом подмассиве по одному элементу, следовательно, подмассивы отсортированы.												
0																		
3																		
1																		
11																		
		Слияние																
4	$3<11$	Сравнение. Условие верно, т.к. сортировка «по возрастанию»																
5	Массив D: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	1	2	3	4	5	6	7	3	0	0	0	0	0	0	0	Следовательно, в массив D записывается число 3.
0	1	2	3	4	5	6	7											
3	0	0	0	0	0	0	0											
6	Массив D: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>11</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	1	2	3	4	5	6	7	3	11	0	0	0	0	0	0	Так как в левой части массива элементы закончены, а в правой остались (число 11), то они все переписываются в массив D. в массив D записывается число 11.
0	1	2	3	4	5	6	7											
3	11	0	0	0	0	0	0											
7	Массив A: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>11</td><td>5</td><td>5</td><td>7</td><td>1</td><td>20</td><td>12</td></tr></table> ←	0	1	2	3	4	5	6	7	3	11	5	5	7	1	20	12	Запись изменений в массив A В подмассиве (3, 11, 5, 5) левая часть (3, 11) отсортирована.
0	1	2	3	4	5	6	7											
3	11	5	5	7	1	20	12											
8	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>11</td><td>5</td><td>5</td><td>7</td><td>1</td><td>20</td><td>12</td></tr></table>	0	1	2	3	4	5	6	7	3	11	5	5	7	1	20	12	Сортируется правая часть (5, 5)
0	1	2	3	4	5	6	7											
3	11	5	5	7	1	20	12											
		Разбиение																
9	$(2+3)/2=2$ левый: <table><tr><td>2</td></tr><tr><td>5</td></tr></table> правый: <table><tr><td>3</td></tr><tr><td>5</td></tr></table>	2	5	3	5	Деление массива (5, 5) на левый под-массив (5 , индекс 2) и правый (5, индекс 3). Границы начала и конца каждого подмассив совпадают. В каждом под-массиве по одному элементу, следовательно, подмассивы отсортированы												
2																		
5																		
3																		
5																		
		Слияние																
10	$5_{\text{(индекс 2)}} < 5_{\text{(индекс 3)}}$	Сравнение. Условие не верно, т.к. сортировка «по возрастанию»																
11	Массив D: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>0</td><td>0</td><td>5</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	1	2	3	4	5	6	7	0	0	5	0	0	0	0	0	Следовательно, в массив D записывается число 5 (индекс 3).
0	1	2	3	4	5	6	7											
0	0	5	0	0	0	0	0											
12	Массив D: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>0</td><td>0</td><td>5</td><td>5</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	1	2	3	4	5	6	7	0	0	5	5	0	0	0	0	Так как в правой части массива элементы закончены, а в левой остались (число 5, индекс 2), то они все переписываются в массив D. В массив D записывается число 5, индекс 2.
0	1	2	3	4	5	6	7											
0	0	5	5	0	0	0	0											
13	Массив A: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>11</td><td>5</td><td>5</td><td>7</td><td>1</td><td>20</td><td>12</td></tr></table> ←	0	1	2	3	4	5	6	7	3	11	5	5	7	1	20	12	Запись изменений в массив A В подмассиве (3, 11, 5, 5) правая часть (5, 5) отсортирована. Подмассивы (3, 11) и (5, 5) отсортированы, следовательно – слияние отсортированных подмассивов.
0	1	2	3	4	5	6	7											
3	11	5	5	7	1	20	12											

	Шаги алгоритма	Описание																															
		Слияние																															
14	<div><table><tr><td></td><td>0</td><td>1</td></tr><tr><td></td><td>3</td><td>11</td></tr><tr><td></td><td>3<5</td><td></td></tr><tr><td></td><td>5</td><td>5</td></tr><tr><td></td><td>2</td><td>3</td></tr></table><p>Массив D:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table></div>		0	1		3	11		3<5			5	5		2	3	0	1	2	3	4	5	6	7	3	0	0	0	0	0	0	0	Сравнение. 3<5 , условие верно, т.к. сортировка «по возрастанию». Следовательно, в D записывается 3 .
	0	1																															
	3	11																															
	3<5																																
	5	5																															
	2	3																															
0	1	2	3	4	5	6	7																										
3	0	0	0	0	0	0	0																										
15	<div><table><tr><td></td><td>0</td><td>1</td></tr><tr><td></td><td>3</td><td>11</td></tr><tr><td></td><td>11<5</td><td></td></tr><tr><td></td><td>5</td><td>5</td></tr><tr><td></td><td>2</td><td>3</td></tr></table><p>Массив D:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>5</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table></div>		0	1		3	11		11<5			5	5		2	3	0	1	2	3	4	5	6	7	3	5	0	0	0	0	0	0	В под-массиве (3, 11) рассматривается следующий элемент (11). Сравнение. 11<5 , условие не верно, т.к. сортировка «по возрастанию». Следовательно, в D записывается 5 .
	0	1																															
	3	11																															
	11<5																																
	5	5																															
	2	3																															
0	1	2	3	4	5	6	7																										
3	5	0	0	0	0	0	0																										
16	<div><table><tr><td></td><td>0</td><td>1</td></tr><tr><td></td><td>3</td><td>11</td></tr><tr><td></td><td>11<5</td><td></td></tr><tr><td></td><td>5</td><td>5</td></tr><tr><td></td><td>2</td><td>3</td></tr></table><p>Массив D:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>5</td><td>5</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table></div>		0	1		3	11		11<5			5	5		2	3	0	1	2	3	4	5	6	7	3	5	5	0	0	0	0	0	В подмассиве (5, 5) рассматривается следующий элемент (5, индекс 3). Сравнение. 11<5 , условие не верно, т.к. сортировка «по возрастанию». Следовательно, в D записывается 5 .
	0	1																															
	3	11																															
	11<5																																
	5	5																															
	2	3																															
0	1	2	3	4	5	6	7																										
3	5	5	0	0	0	0	0																										
17	<div><table><tr><td></td><td>0</td><td>1</td></tr><tr><td></td><td>3</td><td>11</td></tr><tr><td></td><td>11<5</td><td></td></tr><tr><td></td><td>5</td><td>5</td></tr><tr><td></td><td>2</td><td>3</td></tr></table><p>Массив D:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>5</td><td>5</td><td>11</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table></div>		0	1		3	11		11<5			5	5		2	3	0	1	2	3	4	5	6	7	3	5	5	11	0	0	0	0	В под-массиве (5, 5) просмотрены все элементы. В подмассиве (3, 11) элементы остались, следовательно, они все переписываются в массив D. В массив D записывается 11 .
	0	1																															
	3	11																															
	11<5																																
	5	5																															
	2	3																															
0	1	2	3	4	5	6	7																										
3	5	5	11	0	0	0	0																										
18	<div><p>Массив A:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>5</td><td>5</td><td>11</td><td>7</td><td>1</td><td>20</td><td>12</td></tr></table><p>←</p></div>	0	1	2	3	4	5	6	7	3	5	5	11	7	1	20	12	Запись изменений в массив A В массиве (3,5,5,11,7,1,20,12) левая часть (3,5,5,11) отсортирована. Сортируется правая часть (7,1,20,12).															
0	1	2	3	4	5	6	7																										
3	5	5	11	7	1	20	12																										
		Повторяются действия со 2 по 18 для правой части (: 7,1,20,12)																															

	Шаги алгоритма	Описание																																									
19	Массив А: (4+7)/2=5 <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>5</td><td>5</td><td>11</td><td>7</td><td>1</td><td>20</td><td>12</td></tr></table>	0	1	2	3	4	5	6	7	3	5	5	11	7	1	20	12	Деление массива на левый под-массив (элементы с индексами с 4 по 5) и правый под-массив (элементы с индексами с 6 по 7). И так далее.																									
0	1	2	3	4	5	6	7																																				
3	5	5	11	7	1	20	12																																				
20	Массив А: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>3</td><td>5</td><td>5</td><td>11</td><td>1</td><td>7</td><td>12</td><td>20</td></tr></table>	0	1	2	3	4	5	6	7	3	5	5	11	1	7	12	20	В результате в массиве (3,5,5,11,1,7,12,20) левая часть (подмассив 3,5,5,11) и правая часть (подмассив 1,7,12,20) отсортированы. Следовательно – <i>слияние отсортированных под-массивов</i>																									
0	1	2	3	4	5	6	7																																				
3	5	5	11	1	7	12	20																																				
		Слияние																																									
21	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>3</td><td>5</td><td>5</td><td>11</td></tr><tr><td></td><td>3</td><td>1</td><td></td><td></td></tr><tr><td></td><td>1</td><td>7</td><td>12</td><td>20</td></tr><tr><td></td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table> Массив D: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>		0	1	2	3		3	5	5	11		3	1				1	7	12	20		4	5	6	7	0	1	2	3	4	5	6	7	1	0	0	0	0	0	0	0	Сравнение. 3<1 , условие не верно, т.к. сортировка «по возрастанию». Следовательно, в D записывается 1 .
	0	1	2	3																																							
	3	5	5	11																																							
	3	1																																									
	1	7	12	20																																							
	4	5	6	7																																							
0	1	2	3	4	5	6	7																																				
1	0	0	0	0	0	0	0																																				
22	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>3</td><td>5</td><td>5</td><td>11</td></tr><tr><td></td><td>3</td><td>7</td><td></td><td></td></tr><tr><td></td><td>1</td><td>7</td><td>12</td><td>20</td></tr><tr><td></td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table> Массив D: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>3</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>		0	1	2	3		3	5	5	11		3	7				1	7	12	20		4	5	6	7	0	1	2	3	4	5	6	7	1	3	0	0	0	0	0	0	Сравнение. 3<7 , условие верно, т.к. сортировка «по возрастанию». Следовательно, в D записывается 3 .
	0	1	2	3																																							
	3	5	5	11																																							
	3	7																																									
	1	7	12	20																																							
	4	5	6	7																																							
0	1	2	3	4	5	6	7																																				
1	3	0	0	0	0	0	0																																				
23	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>3</td><td>5</td><td>5</td><td>11</td></tr><tr><td></td><td></td><td>5</td><td>7</td><td></td></tr><tr><td></td><td>1</td><td>7</td><td>12</td><td>20</td></tr><tr><td></td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table> Массив D: <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>		0	1	2	3		3	5	5	11			5	7			1	7	12	20		4	5	6	7	0	1	2	3	4	5	6	7	1	3	5	0	0	0	0	0	Сравнение. 5_{индекс1}< 7 , условие верно, т.к. сортировка «по возрастанию». Следовательно, в D записывается 5 с индексом 1.
	0	1	2	3																																							
	3	5	5	11																																							
		5	7																																								
	1	7	12	20																																							
	4	5	6	7																																							
0	1	2	3	4	5	6	7																																				
1	3	5	0	0	0	0	0																																				
24	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>3</td><td>5</td><td>5</td><td>11</td></tr><tr><td></td><td></td><td></td><td>5</td><td>7</td></tr><tr><td></td><td>1</td><td>7</td><td>12</td><td>20</td></tr><tr><td></td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table> Массив D:		0	1	2	3		3	5	5	11				5	7		1	7	12	20		4	5	6	7	Сравнение. 5_{индекс2}< 7 , условие верно, т.к. сортировка «по возрастанию». Следовательно, в D записывается 5 с индексом 2.																
	0	1	2	3																																							
	3	5	5	11																																							
			5	7																																							
	1	7	12	20																																							
	4	5	6	7																																							

	Шаги алгоритма	Описание																																							
	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>5</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	1	2	3	4	5	6	7	1	3	5	5	0	0	0	0																								
0	1	2	3	4	5	6	7																																		
1	3	5	5	0	0	0	0																																		
25	<div><table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>3</td><td>5</td><td>5</td><td>11</td></tr><tr><td></td><td colspan="4">11<7</td></tr></table><table><tr><td>1</td><td>7</td><td>12</td><td>20</td></tr><tr><td>4</td><td>5</td><td>6</td><td>7</td></tr></table><p>Массив D:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>5</td><td>7</td><td>0</td><td>0</td><td>0</td></tr></table></div>		0	1	2	3		3	5	5	11		11<7				1	7	12	20	4	5	6	7	0	1	2	3	4	5	6	7	1	3	5	5	7	0	0	0	<p>Сравнение. 11 < 7 , условие не верно, т.к. сортировка «по возрастанию».</p> <p>Следовательно, в D записывается 7.</p>
	0	1	2	3																																					
	3	5	5	11																																					
	11<7																																								
1	7	12	20																																						
4	5	6	7																																						
0	1	2	3	4	5	6	7																																		
1	3	5	5	7	0	0	0																																		
26	<div><table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>3</td><td>5</td><td>5</td><td>11</td></tr><tr><td></td><td colspan="4">11<12</td></tr></table><table><tr><td>1</td><td>7</td><td>12</td><td>20</td></tr><tr><td>4</td><td>5</td><td>6</td><td>7</td></tr></table><p>Массив D:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>5</td><td>7</td><td>11</td><td>0</td><td>0</td></tr></table></div>		0	1	2	3		3	5	5	11		11<12				1	7	12	20	4	5	6	7	0	1	2	3	4	5	6	7	1	3	5	5	7	11	0	0	<p>Сравнение. 11 < 12 , условие верно, т.к. сортировка «по возрастанию».</p> <p>Следовательно, в D записывается 11.</p>
	0	1	2	3																																					
	3	5	5	11																																					
	11<12																																								
1	7	12	20																																						
4	5	6	7																																						
0	1	2	3	4	5	6	7																																		
1	3	5	5	7	11	0	0																																		
27	<div><table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td></td><td>3</td><td>5</td><td>5</td><td>11</td></tr><tr><td></td><td colspan="4">11<12</td></tr></table><table><tr><td>1</td><td>7</td><td>12</td><td>20</td></tr><tr><td>4</td><td>5</td><td>6</td><td>7</td></tr></table><p>Массив D:</p><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>5</td><td>7</td><td>11</td><td>12</td><td>20</td></tr></table></div>		0	1	2	3		3	5	5	11		11<12				1	7	12	20	4	5	6	7	0	1	2	3	4	5	6	7	1	3	5	5	7	11	12	20	<p>Подмассив (3,5,5,11) просмотрены все элементы</p> <p>В подмассиве (1,7,12,20) элементы остались (12,20), следовательно, они все переписываются в массив D.</p> <p>В массив D записывается 12,20.</p> <p>Массив D отсортирован</p>
	0	1	2	3																																					
	3	5	5	11																																					
	11<12																																								
1	7	12	20																																						
4	5	6	7																																						
0	1	2	3	4	5	6	7																																		
1	3	5	5	7	11	12	20																																		
28	<p>Массив A:</p> <table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>1</td><td>3</td><td>5</td><td>5</td><td>7</td><td>11</td><td>12</td><td>20</td></tr></table>	0	1	2	3	4	5	6	7	1	3	5	5	7	11	12	20	<p>Запись изменений в массив A.</p> <p>Массив A отсортирован.</p>																							
0	1	2	3	4	5	6	7																																		
1	3	5	5	7	11	12	20																																		

Исходный массив – целочисленный массив A.

Дополнительный массив – целочисленный массив D.

L – индекс начала левого (первого) фрагмента;

R - индекс конца правого (второго) фрагмента - конец массива;

Фрагмент программного кода «Сортировка слиянием»:

```
static void SL(int L, int SR, int R, int[] A)
{
    //L - начало левого(первого) фрагмента;
    //SR -начало правого(второго) фрагмента;
    //R - конец правого(второго) фрагмента - конец массива;
    int KL = SR - 1;           //KL - конец левого(первого) фрагм
    int t = L;                 // t=лев. граница первого фрагмента, при первом слиянии t=0;
    //при втором слиянии t=2; при слиянии с индекса 0 по индекс 3, t=0
    int[] D = new int[A.Length]; // Размер массив D равен размеру массива A
    int num_el;                // количество эл-ов для слияния
    num_el = (R - L + 1);
    // пока не закончен хотя бы один из фрагментов (KL -конец левого фр.,R-конец
    //правого фрвгмента), элементы записываются из массива A в массив D
    while ((L <= KL) && (SR <= R))
    {
        if (A[L] <= A[SR])
        {
            D[t] = A[L]; L++; t++; //запись в массив D элементов из левой части массива
        }
        else
        {
            D[t] = A[SR]; SR++; t++; //запись в массив D элементов из правой части массива
        }
    } // конец цикла While
    //Если один фрагмент обработан полностью, а во втором остались элементы, то
    //в массив D переносится остаток не обработанного до конца фрагмента
    while (L <= KL)
    {
        D[t] = A[L]; L++; t++; // в D переносится остаток из левого фрагмента
    }
    while (SR <= R)
    {
        D[t] = A[SR]; SR++; t++; // в D переносится остаток из правого фрагмента
    }
    //Запись изменений в массив A (массив не переписывается)
    for (int j = 0; j < num_el; j++) // после первого слияния в массиве A изменились
    {                               //первые два эл-та
        A[R] = D[R]; R--; }
    }
```

Метод сортировки:

```
static void Sort(int L, int R, int[] A)
{
    if (R > L)
    {
        int SR = (R + L) / 2; // определение середины массива
        Sort(L, SR, A);      // рекурсивное обращение к разбиению левой части массива
        Sort((SR + 1), R, A); // рекурсивное обращение к разбиению правой части массива
        SL(L, (SR + 1), R, A); // слияние отсортированных частей массива
    }
}
```

Задания для выполнения в аудитории

Задание №1. Реализуйте алгоритм сортировки «Простым слиянием» для целочисленного массива A из N элементов. Массив заполнен «случайным образом»; количество элементов задается константой.

Рекомендации к выполнению задания:

В методе Main необходимо выполнить следующие действия:

- задать количество элементов массива;
- описать и инициализировать массив A;
- вывести на экран массив A до сортировки;
- вызвать метод Sort с параметрами:
 - L – индекс начала левого (первого) фрагмента;
 - R – индекс конца правого (второго) фрагмента – конец массива;
 - A – исходный массив;
- вывести на экран массив A после сортировки.

Задания для самостоятельной работы студента (СРС)

Задание №1. Для Задания №1 из аудиторной работы: внесите дополнения в программу, которые позволят вывести на экран *каждый из фрагментов* массива и *результат слияния фрагментов*, в процессе выполнения алгоритма слияния.

СОРТИРОВКА ХОАРА (Быстрая сортировка)

Быстрая сортировка, часто называемая qsort – широко известный алгоритм сортировки, разработанный Чарльзом Хоаром в 1960 г. Один из быстрых, универсальных алгоритмов сортировки массива (в среднем $O(n \log n)$) обменов.

Алгоритм (сортировка по возрастанию): менять местами далеко стоящие друг от друга элементы. Для этого:

1. Выбрать для сравнения один элемент X (средний; барьерный, опорный).
2. Найти слева от X первый элемент, который (больше) X .
3. Найти справа от X первый элемент, который (меньше) X .
4. Найденные элементы поменять местами.

После *первого* прохода все элементы, которые меньше X , будут стоять слева от X , а все элементы, которые больше X , – справа от X .

Для каждой из половинок массива:

- определяют барьерный элемент;
- осуществляют перестановку элементов относительно вновь определенного барьерного элемента.

Деление этих половин и перестановку элементов продолжают до тех пор, пока не останется в них по 1 элементу.

Достоинства:

- Один из самых быстросортирующих из алгоритмов внутренней сортировки общего назначения.
- Прост в реализации.
- Требуется $O(\log_2 n)$ дополнительной памяти для своей работы.
- Существует эффективная модификация для сортировки строк.

Недостатки:

- Увеличение скорости при неудачных выборах опорных элементов.
- Реализация алгоритма может привести к ошибке переполнения стека, так как ей может потребоваться сделать $O(n)$ вложенных рекурсивных вызовов.
- Неустойчив

Пример работы алгоритма для массива из 8 элементов:

Таблица 1

Индекс элемента	0	1	2	3	4	5	6
Элемент массива	13	8	14	12	7	4	1

Таблица 2

–	0	1	2	3	4	5	6
–	13	8	14	12	7	4	1

Таблица 3

–	0	1	2	3	4	5	6
–	13	8	14	12	7	4	1

Таблица 4

–	0	1	2	3	4	5	6
–	1	8	14	12	7	4	13

Таблица 5

–	0	1	2	3	4	5	6
–	1	8	14	12	7	4	13

Таблица 6

–	0	1	2	3	4	5	6
–	1	8	14	12	7	4	13

Таблица 7

–	0	1	2	3	4	5	6
–	1	8	4	12	7	14	13

Таблица 8

–	0	1	2	3	4	5	6
–	1	8	4	12	7	14	13

Таблица 9

–	0	1	2	3	4	5	6
–	1	8	4	12	7	14	13

Таблица 10

–	0	1	2	3	4	5	6
–	1	8	4	7	12	14	13

Таблица 11

–	0	1	2	3	4	5	6
–	1	8	4	7	12	14	13

Таблица 12

–	0	1	2	3	4	5	6
–	1	8	4	7	12	14	13

Таблица 13

–	0	1	2	3	4	5	6
–	1	8	4	7	12	14	13

В лев части среди 1 И 8 эл-ов нет. (сам эл-т =8)

Таблица 14

–	0	1	2	3	4	5	6
–	1	7	4	8	12	14	13

Таблица 15

–	0	1	2	3	4	5	6
–	1	7	4	8	12	14	13

Таблица 16

–	0	1	2	3	4	5	6
–	1	7	4	8	12	14	13

Таблица 17

–	0	1	2	3	4	5	6
–	1	4	7	8	12	14	13

Таблица 18

–	0	1	2	3	4	5	6
–	1	4	7	8	12	14	13

Таблица 19

–	0	1	2	3	4	5	6
–	1	4	7	8	12	14	13

Таблица 20

–	0	1	2	3	4	5	6
–	1	4	7	8	12	13	14

Фрагмента программного кода:

LG – левая граница; PG – правая граница; $X = M[(LG + PG) / 2]$ – вычисление барьерного элемента

Фрагмента программного кода с комментариями:

```
static void QSort(int[] M,int LG,int PG)
```

```
{
    int i = LG, j = PG;
    // Находим разделительный элемент в середине массива
    double X = M[(LG + PG) / 2];
    // Обход массив
    while (i <= j)
    {
        //Находим элемент, который больше или равен разделительному элементу от //левого
        //индекса.
        while (M[i] < X) ++i;
        // Находим элемент, который меньше или равен разделительному элементу от
        //правого индекса.
        while (M[j] > X) --j;
        // Если индексы не пересекаются, меняем элементы
        if (i <= j)
        {
            int T; T = M[i]; M[i] = M[j]; M[j] = T;
            ++i; --j;
        } //if
    } //while

    // если правый индекс не достиг левой границы массива,то нужно повторить
    //сортировку левой части.
    if (LG < j) QSort(M,LG,j);
    //если левый индекс не достиг правой границы массива,то нужно повторить
    // сортировку правой части.
    if (i < PG) QSort(M, i,PG);
} // QSort
```


Задания для выполнения в аудитории

Задание №1. Реализуйте алгоритм быстрой сортировки по возрастанию для целочисленного одномерного массива.

Требования к программе:

- Размерность массива определите в разделе констант.
- Заполнение массива – «случайным образом».
- Обеспечьте вывод:
 - исходного массива после заполнения;
 - всего массива после каждого обмена элементов;
 - значения разделительного («среднего» элемента) массива;
 - всего массива после сортировки;
- **LG** – левая граница массива; **PG**- правая граница массива.

Задания для самостоятельной работы студента (СРС)

Задание №1. Используя программный код аудиторной работы, дополните его необходимыми командами для:

- заполнения массива «лучшим» и «худшим» способом;
- вывода на экран *количества сравнений и перестановок* для заполнения массива следующими способами: «лучшим», «худшим» и «случайным образом».

Задание №2. Дан массив КТ, состоящий из координат точек А, В, С, ... и т. д.. Количество точек: N>10.

Например, есть точки А(7; 2), В(46; 5),... С(3; 5), тогда массив КТ состоит из элементов: 7; 2; 46; 5;.... 3; 5.

Необходимо упорядочить по возрастанию точки *по x-координате*. Для указанного массива, например: 3; 5; 7; 2;.... 46; 5.

Требования к программе:

- N – вводится с клавиатуры.
- Способ заполнения массива – на усмотрение автора программы.
- Использовать «Быструю сортировку».
- Вывод исходного и упорядоченного массива.

МНОГОМЕРНЫЕ МАССИВЫ

- **ДВУМЕРНЫЕ МАССИВЫ**
- **МАССИВЫ ТРЕХ И БОЛЕЕ ИЗМЕРЕНИЙ**

Трехмерный целочисленный массив M3 размерами 2х3х4 – 24 элемента.

```
int [,,] M3 = new int [2,3,4];
```

- **СТУПЕНЧАТЫЕ МАССИВЫ**

Ступенчатые массивы - специальный тип двумерного массива, представляющий собой массив массивов, в котором длина каждого массива может быть разной.

```
int[][] styp = new int[3][]; // Объявление двумерного ступенчатого массива styp. Первая  
размерность =3.
```

```
    styp[0] = new int[4];
```

```
    styp[1] = new int[3];
```

```
    styp[2] = new int[5];
```

Пример заполненного ступенчатого массива styp:

```
6 8 4 1
```

```
7 2 4
```

```
1 0 3 4 8
```

ДВУМЕРНЫЕ МАССИВЫ

СПОСОБЫ ОРГАНИЗАЦИИ МАССИВА

1. **Без инициализации** (объявление с отложенной инициализацией)

int [,] RITM – объявление целочисленного двумерного массива RITM с отложенной инициализацией.

2. **Объявление массива с инициализацией.**

2.1. Инициализация массива при его создании (явная инициализация)

Общая форма объявления двумерного массива с инициализации:

```
тип[,] имя_массива = { {v1, v2, v3}, {v4, v5, v6}, ..., } ;
```

Пример 1. Объявление целочисленного двумерного массива W [2,4] с явной инициализацией.

Способ 1:

```
int[,] W = {  
    { 10, 11, 12, 13},  
    { 14, 15, 16, 10}  
};
```

Способ 2:

```
int [,] W = { { 10, 11, 12, 13}, { 14, 15, 16, 10} };
```

2.2. Инициализация массива в объектном стиле (инициализация массива с использованием конструктора)

Общая форма для объявления двумерного массива в объектном стиле:

```
тип[,] имя_массива = new тип[n,m] ;
```

где **тип** определяет тип элементов массива; **n** – число строк, **m** – число столбцов в массиве.

Объявление целочисленного двумерного массива размером [4,8] в объектном стиле:

```
int[,] pr = new int[4,8];
```

ДЕЙСТВИЯ НАД ЭЛЕМЕНТАМИ МАССИВА

1. Заполнение массива

1.1. Ввод с клавиатуры

1.2. «Случайным образом»

1.3. По формуле

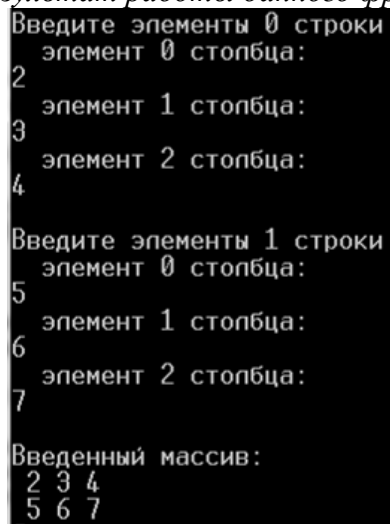
1.4. Из текстового файла

1.1. Ввод с клавиатуры

Пример 2. Фрагмент программного кода для ввода шести элементов двумерного массива pr[2,3]. Элементы массива вводятся «по строкам».

```
for (int i = 0; i < 2; i++)  
{  
    Console.WriteLine("Введите элементы " + i + " строки ");  
    for (int j = 0; j < 3; j++)  
    {  
        Console.WriteLine(" элемент " + j + " столбца: ");  
        pr[i, j] = Convert.ToInt32(Console.ReadLine());  
    }  
    Console.WriteLine();  
}
```

Результат работы данного фрагмента:



```
Введите элементы 0 строки  
элемент 0 столбца:  
2  
элемент 1 столбца:  
3  
элемент 2 столбца:  
4  
Введите элементы 1 строки  
элемент 0 столбца:  
5  
элемент 1 столбца:  
6  
элемент 2 столбца:  
7  
Введенный массив:  
2 3 4  
5 6 7
```

Ввод элементов двумерного массива «по столбцам» см. Приложение 1

1.2. «Случайным образом»

Пример 3. Фрагмент программного кода для заполнения двумерного массива pr с использованием генератора случайных чисел (R - объект класса Random)

```
int[,] pr = new int[5,3];  
for (int i = 0; i < 5; i++)  
for (int j = 0; j < 3; j++) pr[i,j] = R.Next(10, 20);
```

Результат работы данного фрагмента:

16 17 10
15 13 15
11 15 12
17 15 11
13 18 14

1.3. По формуле

Пример 4. Заполнение двумерного массива `pr` по формуле $i * 2$.

Фрагмент программного кода:

```
int[,] pr = new int[2,4];  
for (int i = 0; i < 2; i++)  
    for (int j = 0; j < 4; j++) pr[i,j] = pr[i,j] = i * 2;
```

Результат работы данного фрагмента:

Заполненный массив:

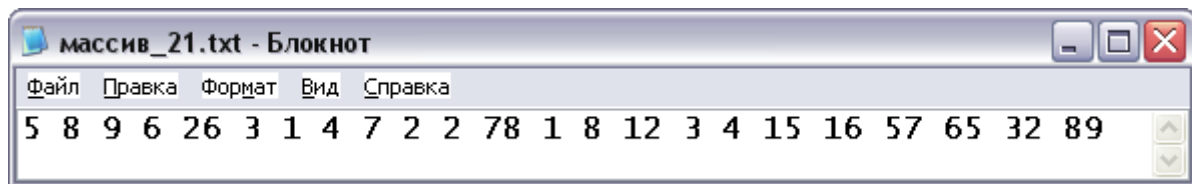
0 0 0 0
2 2 2 2

1.4. Из текстового файла

Заполнение целочисленного двумерного массива `pr` значениями из текстового файла `массив_21.txt`

Дано:

– текстовый файл `массив_21.txt`:



Элементы файла записаны в одну строку, через пробел.

– описан целочисленный двумерный массив `pr`.

Требуется: заполнить массив `pr` из файла `массив_2.txt`

Решение:

1. Чтение из файла в строковую переменную:

- создать объект, например, `f2` класса `StreamReader`:

```
StreamReader f2 = new StreamReader("D:/массив_2.txt");
```

- прочесть из файла символы как одну строку в строковую переменную `S` (метод `ReadToEnd()`):

```
string S=f1.ReadToEnd();
```

Результат: в переменной `S` хранятся символы 5 8 9 6 26 3 1 4 7 2 2 78 1 8 12 3 4 15 16 57 65 32 89.

5. Создается строковый массив `s1` из символов строки `S`, с помощью метода `Split(' ')`. Метод `Split()` разделяет заданную строку на подстроки, в качестве разделителя использует символ, указанный в качестве параметра.

```
string[] s1 = S.Split(' ');
```

6. Преобразование строкового массива `s1` в целочисленный двумерный массив `pr`:

```
int k = 0; // индекс строкового массива s1
```

```

for (i = 0; i < 3; i = i + 1)
{
    for (j = 0; j < 4; j++)
    {
        pr[i, j] = Convert.ToInt32(s1[k]
        k = k + 1; // изменение индекса строкового массива s1
    }
}

```

Пример 5. Заполнение целочисленного двумерного массива pr значениями из текстового файла «массив_21.txt»

Фрагмент программного кода:

```

int[,] pr = new int[3, 4];
int i, j;
StreamReader f2 = new StreamReader("D:/массив_21.txt");
string S = f2.ReadToEnd(); // считывание потока до конца
string[] s1 = S.Split(' '); // преобразование в строковый массив s1

int k = 0; // индекс строкового массива s1
for (i = 0; i < 3; i = i + 1)
{
    for (j = 0; j < 4; j++)
    {
        pr[i, j] = Convert.ToInt32(s1[k]); // запись "по строкам" в двумерный массив pr,
        // элементов из строкового массива s1. При этом каждый символ строкового массива s1
        // конвертируется в целый тип.
        k = k + 1; // изменение индекса строкового массива s1
    }
}

```

Результат работы данного фрагмента:

5	8	9	6
26	3	1	4
7	2	2	78

Для продолжения нажмите любую клавишу

Второй вариант заполнения двумерного массива из файла см. Приложение 2

2. Вывод элементов двумерного массива на экран, в виде таблицы

Пример 6. Вывод на экран двумерного массива pr с использованием цикла for
Фрагмент программного кода для вывода на экран целочисленного массива pr.

```

int[,] pr = new int[3, 4];
for (i = 0; i < 3; i = i + 1) // задается номер строки
{
    for (j = 0; j < 4; j = j + 1) // задается номер столбца
        Console.Write("t" + pr[i, j]); // вывод с использованием управляющего символа \t
    Console.WriteLine();
}

```

3. Обработка отдельно строк и столбцов

3.1. Обработка отдельной строки

Пример 7. Вывод на экран строки с индексом 2:

Исходный массив:

5	7	8	9
6	3	1	4
2	2	78	1

Фрагмент программного кода для вывода на экран строки с индексом 2:

```
i = 2; //индекс выводимой строки
Console.WriteLine("Элементы строки с индексом " + i + ":");
for (j = 0; j < 4; j++) Console.WriteLine(" " + pr[i, j]);
```

Результат работы данного фрагмента:

Элементы строки с индексом 2: 2 2 78 1

3.2. Обработка отдельного столбца

Пример 8. Вывод на экран столбца с индексом 1:

Исходный массив:

5	7	8	9
6	3	1	4
2	2	78	1

Фрагмент программного кода для вывода на экран строки с индексом 2:

```
j = 1; //индекс выводимого столбца
Console.WriteLine("Элементы столбца с индексом " + j + ":");
for (i = 0; i < 3; i++) Console.WriteLine(" " + pr[i, j]);
```

Результат работы данного фрагмента:

Элементы столбца с индексом 1: 7 3 2

3.3. Обмен строк

Алгоритм обмена строк с индексом R и T, с использованием временной переменной WR:

1. Определяется временная переменная, например WR (тип WR совпадает с типом элементов массива)
2. Просматривая все элементы строки T:
 - в WR помещается элемент строки T;
 - на место элемента строки T помещается соответствующий элемент строки R;
 - на место элемента строки R перемещается элемент из переменной WR.

Пример 9. Обмен строк с индексами 0 и 2.

Исходный массив:

		Индекс столбца j			
		0	1	2	3
Индекс строки i	0	5	7	8	9
	1	6	3	1	4
	2	2	2	78	1

Фрагмент программного кода:

```
int WR;
for (j = 0; j < 4; j++)
{
    WR = pr[0, j];
```

Результат работы данного фрагмента:

2	2	78	1
6	3	1	4
5	7	8	9

<pre> pr[0,j]=pr[2,j]; pr[2,j]= WR; } </pre>	
--	--

3.4. Обмен столбцов

Алгоритм обмена столбцов аналогичен алгоритму обмена строк.

Пример 10. Обмен столбцов с индексами 1 и 3.

Исходный массив:

		Индекс столбца j			
		<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
Индекс строки i	<i>0</i>	5	7	8	9
	<i>1</i>	6	3	1	4
	<i>2</i>	2	2	78	1

Фрагмент программного кода:	Результат работы данного фрагмента:												
<pre>int WR; for (i = 0; i < 3; i++) { WR =pr[i,1]; pr[i,1]=pr[i,3]; pr[i,3]= WR; }</pre>	<table><tr><td>5</td><td>9</td><td>8</td><td>7</td></tr><tr><td>6</td><td>4</td><td>1</td><td>3</td></tr><tr><td>2</td><td>1</td><td>78</td><td>2</td></tr></table>	5	9	8	7	6	4	1	3	2	1	78	2
5	9	8	7										
6	4	1	3										
2	1	78	2										

Поиск в двумерном массиве, передача массива в метод осуществляется аналогично одномерному массиву.

Задания для выполнения в аудитории

Задание №1. Разработайте консольное приложение для работы с двумерным массивом вещественных чисел.

Выполните над элементами массива следующие действия:

1. Заполните массив «случайным образом».
2. Выведите элементы исходного двумерного массива в виде таблицы.
3. Найдите сумму элементов каждой строки. Значения сумм каждой строки выведите в следующем виде, например:

$$S_0=12$$

$$S_1=\dots$$

4. Выведите на экран элементы N-го столбца. Значение N введите с клавиатуры.
5. Поменяйте местами элементы строки с индексами K1 и K2, K1 и K2 введите с клавиатуры. Выведите на экран элементы массива после обмена строк в виде таблицы.

Задания для самостоятельной работы студента (СРС)

Задание №1. Выполните задание, согласно Вашему варианту.

Для всех вариантов: заполнение двумерного массива из текстового файла.

Вариант	Задание
1	Дана целочисленная матрица (N x M). Определить: 1) количество строк, не содержащих ни одного нулевого элемента; 2) максимальное из чисел, встречающихся в заданной матрице более одного раза.
2	Дана целочисленная матрица (N x M). Определить: 1) количество столбцов, содержащих хотя бы один нулевой элемент; 2) номер строки, в которой находится самая длинная серия одинаковых элементов.
3	Дана целочисленная матрица (N x M). Определить: 1) произведение элементов в тех строках, которые не содержат отрицательных элементов; 2) максимум среди сумм элементов 3-ей строки и 5-го столбца.
4	Дана целочисленная матрица (N x M). Определить: 1) сумму элементов в тех столбцах, которые не содержат отрицательных элементов; 2) минимум среди сумм элементов каждой строки матрицы.
5	Дана целочисленная матрица (N x M). Определить: 1) номер первого из столбцов, содержащих хотя бы один нулевой элемент; 2) сумму всех отрицательных нечетных элементов.
6	Дана целочисленная матрица (N x M). Определить: 1) количество одинаковых элементов в каждой строке; 2) номер первого из столбцов, не содержащих ни одного отрицательного элемента.
7	Дана целочисленная матрица (N x M). Определить: 1) сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент;

Вариант	Задание
	2) произведение элементов в тех столбцах, которые содержат хотя бы один положительный элемент.
8	Дана целочисленная матрица ($N \times M$). Определить: 1) сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент; 2) количество элементов кратных четырем в k -м столбце (k вводится с клавиатуры).
9	Дана целочисленная матрица ($N \times M$). Определить: 1) сумму элементов в тех строках, которые содержат хотя бы один элемент, равный M (M – введите с клавиатуры); 2) произведение элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.
10	Дана целочисленная матрица ($N \times M$). Определить : 1) количество столбцов, не содержащих ни одного нулевого элемента; 2) сумму всех положительных четных элементов.
11	Дана целочисленная матрица ($N \times M$). Определить: 1) количество столбцов, не содержащих ни одного элемента, равного A ; 2) минимальное из чисел, встречающихся в заданной матрице более одного раза.
12	Дана целочисленная матрица ($N \times M$). Определить: 1) номер первой из строк, содержащей хотя бы один нулевой элемент; 2) сумму всех отрицательных четных элементов матрицы.
13	Дана целочисленная матрица ($N \times M$). Определить: 1) количество столбцов, не содержащих ни одного нулевого элемента; 2) минимальное из чисел, встречающихся в заданной матрице более одного раза.
14	Дана целочисленная матрица ($N \times M$). Определить: 1) сумму элементов в тех столбцах, которые содержат хотя бы один элемент, превышающий D ; 2) номер строк, в которых элементы превышают число K (K вводится с клавиатуры). Вывести эти элементы на экран.
15	Дана целочисленная матрица ($N \times M$). Определить: 1) произведение элементов в тех столбцах, которые содержат отрицательные элементы; 2) минимум среди сумм элементов второй строки и четвертого столбца.

Задание №2. Выполните задание, согласно Вашему варианту.

Для каждого задания: разработайте консольное приложение, содержащее *отдельный* класс для выполнения действий над элементами квадратной матрицей. Обязательные члены класса: поля, методы заполнения и вывода матрицы.

Вариант	Задание
1	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «случайным образом»; 2) вывод матрицы на экран в виде таблицы; 3) вычисление произведения элементов главной диагонали; 4) поиск индексов наибольшего элемента среди всех четных строк.
2	<p>Действия над элементами матрицы вещественных чисел:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «случайным образом»; 2) вывод матрицы на экран в виде таблицы; 3) вычисление суммы элементов, расположенных выше главной диагонали; 4) поиск наибольшего элемента среди всех четных столбцов.
3	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы: элементы главной диагонали = 1; элементы выше главной диагонали = -1; элементы ниже главной диагонали = 0; 2) вывод матрицы на экран в виде таблицы; 3) вывод на экран элементов главной диагонали; 4) вычисление суммы элементов главной диагонали; 5) подсчет количества элементов ниже главной диагонали.
4	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «случайным образом»; предусмотреть при заполнении «случайным образом» - наличие отрицательных чисел; 2) вывод матрицы на экран в виде таблицы; 3) обмен местами строки с первым отрицательным элементом со строкой с индексом L, L вводится с клавиатуры; вывести на экран матрицу после обмена. 4) вывод на экран столбца с индексом T, T вводится с клавиатуры.
5	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «по формуле», формулу определите самостоятельно; 2) вывод матрицы на экран в виде таблицы; 3) вычисление произведения элементов главной диагонали; 4) поиск индексов наименьшего элемента среди всех четных столбцов.
6	<p>Действия над элементами матрицы вещественных чисел:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «случайным образом»; 2) вывод матрицы на экран в виде таблицы; 3) вычисление суммы элементов, расположенных ниже главной диагонали; 4) поиск наименьшего элемента среди всех нечетных столбцов.
7	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы: элементы главной диагонали = 0, остальные элементы заполняются «случайным образом»; 2) вывод матрицы на экран в виде таблицы; 3) вывод на экран элементов побочной диагонали; 4) обмен местами минимального и максимального элементов, расположенных не на главной диагонали.
8	<p>Действия над элементами матрицы вещественных чисел:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «случайным образом»; 2) вывод матрицы на экран в виде таблицы; 3) вычисление суммы тех элементов матрицы, которые расположены выше побочной диагонали и имеют четные значения; 4) вывод на экран элементов N-ого столбца, N вводится с клавиатуры.

Вариант	Задание
9	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «случайным образом»; 2) вывод матрицы на экран в виде таблицы; 3) вычисление произведения элементов побочной диагонали; 4) поиск индексов наибольшего элемента среди всех нечетных строк.
10	<p>Действия над элементами матрицы вещественных чисел:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «по формуле», формулу определите самостоятельно; 2) вывод матрицы на экран в виде таблицы; 3) вычисление суммы элементов, расположенных ниже побочной диагонали; 4) поиск наибольшего элемента среди всех нечетных столбцов.
11	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы: элементы главной диагонали = -1; элементы выше главной диагонали по формуле: $i+1$; элементы ниже главной диагонали = 0; 2) вывод матрицы на экран в виде таблицы; 3) сформируйте одномерный массив из элементов, расположенных <i>выше</i> главной диагонали, которые распределятся в одномерном массиве следующим образом: вначале элементы из нулевой строки квадратной матрицы, затем из первой строки квадратной матрицы и т.д..
12	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «по формуле», формулу определите самостоятельно; 2) вывод матрицы на экран в виде таблицы; 3) вычисление произведения элементов побочной диагонали; 4) поиск индексов наименьшего элемента среди всех нечетных столбцов.
13	<p>Действия над элементами матрицы вещественных чисел:</p> <ol style="list-style-type: none"> 1) заполнение матрицы: элементы побочной диагонали = 0; элементы выше побочной диагонали – «случайным образом» (предусмотреть при заполнении «случайным образом» - наличие отрицательных чисел); остальные элементы = -1; 2) вывод матрицы на экран в виде таблицы; 3) вывод на экран элементов матрицы с отрицательными значениями; 4) подсчет количества элементов с отрицательными значениями; 5) вычисление суммы всех элементов матрицы.
14	<p>Действия над элементами матрицы вещественных чисел:</p> <ol style="list-style-type: none"> 1) заполнение матрицы «случайным образом»; 2) вывод матрицы на экран в виде таблицы; 3) вычисление суммы элементов, расположенных выше побочной диагонали; 4) подсчет в матрице количества элементов, больших числа К, К вводится с клавиатуры.
15	<p>Действия над элементами целочисленной матрицы:</p> <ol style="list-style-type: none"> 1) заполнение матрицы: элементы побочной диагонали = 1, остальные элементы «случайным образом»; предусмотреть при заполнении «случайным образом» - наличие отрицательных чисел; 2) вывод матрицы на экран в виде таблицы; 3) сформируйте одномерный массив из <i>положительных</i> элементов, квадратной матрицы; элементы в одномерном массиве должны распределиться следующим образом: вначале элементы из нулевой строки квадратной матрицы, затем из первой строки квадратной матрицы и т.д..

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Левитин А. В. Алгоритмы: введение в разработку и анализ. : Пер. с англ. — М. : Издательский дом Вильяме, 2006
2. Павловская Т.А. С#. Программирование на языке высокого уровня:учебник для вузов. - СПб: Питер, 2009.
3. Сайт о программировании. URL: <https://metanit.com/sharp/tutorial/2.1.php> - METANIT.COM

Приложение 1

Ввод элементов двумерного массива «по столбцам»

Фрагмент кода для ввода элементов двумерного массива по столбцам»

```
for (int j = 0; j < 3; j++)  
{  
    Console.WriteLine("Введите элементы " + j + " столбца ");  
    for (int i = 0; i < 2; i++)  
    {  
        Console.WriteLine(" элемент " + i + " строки: ");  
        pr[i, j] = Convert.ToInt32(Console.ReadLine());  
    }  
    Console.WriteLine();  
}
```

Результат работы фрагмента программного кода:

Введите элементы 0 столбца

элемент 0 строки:

1

элемент 1 строки:

2

Введите элементы 1 столбца

элемент 0 строки:

3

элемент 1 строки:

4

Введите элементы 2 столбца

элемент 0 строки:

5

элемент 1 строки:

6

Введенный массив:

1 3 5

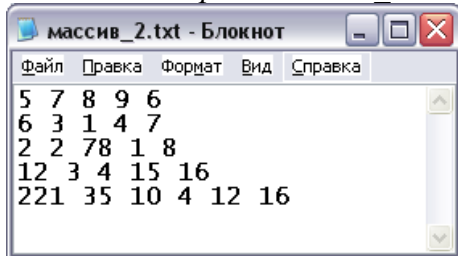
2 4 6

Приложение 2

Заполнение двумерного массива из файла.

Дано:

– текстовый файл *массив_2.txt*:



Элементы массива записаны в несколько строк. Количество строк и столбцов должно быть не меньше, чем размерность двумерного массива, который будет заполняться элементами из текстового файла.

– описан целочисленный двумерный массив *pr*.

Требуется: заполнить массив *pr* из файла *массив_2.txt*

Решение:

1. Чтение из файла в строковую переменную:

- создать объект, например, *f2* класса *StreamReader*:

```
StreamReader f2 = new StreamReader("D:/массив_2.txt");
```

- прочесть из файла символы «построчно», используя метод *ReadLine()*

```
string S = f2.ReadLine();
```

Результат: после считывания из файла первой строки, в переменной *S* хранятся следующие символы: 5 7 8 9 6.

2. Создается *строковый массив* *s1* из символов строки *S*, с помощью метода *Split(' ')*.

Метод *Split()* разделяет заданную строку на подстроки, в качестве разделителя использует символ, указанный в качестве параметра.

```
string[] s1 = S.Split(' ');
```

3. Преобразование *строкового массива* *s1* в *целочисленный двумерный массив* *pr*:

```
for (i = 0; i < 3; i++)
{
    string S = f2.ReadLine();
    string[] s1 = S.Split(' ');

    for (j = 0; j < 4; j++)
        pr[i, j] = Convert.ToInt32(s1[j]);
}
```

Фрагмент программного кода:

```
int[, ] pr = new int[3, 4];
StreamReader f2 = new StreamReader("D:/массив_2.txt");
int i, j;

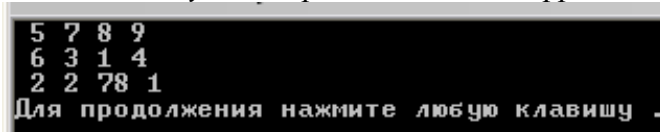
for (i = 0; i < 3; i++)
```

```

{
    string S = f2.ReadLine();// считывание строки символов из потока
    string[] s1 = S.Split(' '); // преобразование в строковый массив s1
    for (j = 0; j < 4; j++) pr[i, j] = Convert.ToInt32( s1[j]); //запись "по строкам"в
//двумерный массив pr, элементов из строкового массива s1. При этом каждый символ
//строкового массива s1 конвертируется в целый тип.
}

```

Результат работы данного фрагмента:



```

5 7 8 9
6 3 1 4
2 2 78 1
Для продолжения нажмите любую клавишу .

```

Количество элементов в строках и столбцах определяется размерностью заданного массива, в данном примере, массива pr[3,4]: `int[,] pr = new int[3, 4];` .

Учебное издание

Ромашкина Татьяна Витальевна
Миндоров Николай Иванович

**Алгоритмизация и основы программирования.
Основы объектно-ориентированного программирования. С#**

Учебное пособие

Редактор *Л. Л. Савенкова*
Корректор *Л. Л. Соболева*
Компьютерная верстка: *Т. В. Ромашкина*

Объем данных 1,25 Мб
Подписано к использованию 27.08.2021

Размещено в открытом доступе
на сайте www.psu.ru
в разделе НАУКА / Электронные публикации
и в электронной мультимедийной библиотеке ELiS

Издательский центр
Пермского государственного
национального исследовательского университета
614990, г. Пермь, ул. Букирева, 15