Л а б о р а т о р н а я р а б о т а № 9

**ПРИЛОЖЕНИЯ WINDOWS FORMS.** [[**РАБОТА С ДВУМЕРНЫМИ МАССИВАМИ**](#_Toc290318461)**.**](#_Toc290318445) **МЕТОДЫ, ПЕРЕГРУЗКИ МЕТОДОВ.**

**Цель работы:**

* ознакомиться с понятиями метода, сигнатур методов, параметров и аргументов метода
* получить навыки в организации [доступа к методу](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/methods#method-access), [передачи параметров по ссылке и передачи по значению](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/methods#passing-by-reference-vs-passing-by-value)
* научиться конструировать и [использовать](https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/methods#return-values) перегруженные методы

**Краткие теоретические сведения.**

# **Методы**

Если переменные хранят некоторые значения, то методы содержат собой набор инструкций, которые выполняют определенные действия. По сути метод - это именованный блок кода, который выполняет некоторые действия.

Общее определение методов выглядит следующим образом:

|  |
| --- |
| [модификаторы] тип\_возвращаемого\_значения название\_метода ([параметры])  {      // тело метода  } |

Модификаторы и параметры необязательны.

Ранее мы уже использовали как минимум один метод - Console.WriteLine(), который выводит информацию на консоль. Теперь рассмотрим, как мы можем создавать свои методы.

# **Определение метода**

Определим один метод:

|  |
| --- |
| void SayHello()  {      Console.WriteLine("Hello");  } |

Здесь определен метод SayHello, который выводит некоторое сообщение. К названиям методов предъявляются в принципе те же требования, что и к названиям переменных. Однако, как правило, названия методов начинаются с большой буквы.

Перед названием метода идет возвращаемый тип данных. Здесь это тип void, который указыает, что фактически ничего не возвращает, он просто производит некоторые действия.

После названия метода в скобках идет перечисление параметров. Но в данном случае скобки пустые, что означает, что метод не принимает никаких параметров.

После списка параметров в круглых скобках идет блок кода, который представляет набор выполняемых методом инструкций. В данном случае блок метода SayHello содержит только одну инструкцию, которая выводит строку на консоль:

|  |
| --- |
| Console.WriteLine("Hello"); |

Но если мы запустим данный проект, то мы не увидим никакой строки, которую должен выводить метод SayHello. Потому что после определения метод еще надо вызвать, чтобы он выполнил свою работу.

# **Вызов методов**

Чтобы использовать метод SayHello, нам надо его вызвать. Для вызова метода указывается его имя, после которого в скобках идут значения для его параметров (если метод принимает параметры).

|  |
| --- |
| название\_метода (значения\_для\_параметров\_метода); |

Например, вызов метода SayHello будет выглядеть следующим образом:

|  |
| --- |
| SayHello(); |

Поскольку метод не принимает никаких параметров, то после названия метода идут пустые скобки.

Сокращенная запись методов

Если метод в качестве тела определяет только одну инструкцию, то мы можем сократить определение метода. Например, допустим у нас есть метод:

|  |
| --- |
| void SayHello()  {      Console.WriteLine("Hello");  } |

Мы можем его сократить следующим образом:

|  |
| --- |
| void SayHello() => Console.WriteLine("Hello"); |

То есть после списка параметров ставится оператор =>, после которого идет выполняемая инструкция.

# **Перегрузка методов**

Иногда возникает необходимость создать один и тот же метод, но с разным набором параметров. И в зависимости от имеющихся параметров применять определенную версию метода. Такая возможность еще называется перегрузкой методов (method overloading).

И в языке C# мы можем создавать в классе несколько методов с одним и тем же именем, но разной сигнатурой. Что такое сигнатура? Сигнатура складывается из следующих аспектов:

* Имя метода
* Количество параметров
* Типы параметров
* Порядок параметров
* Модификаторы параметров

Но названия параметров в сигнатуру **НЕ** входят. Например, возьмем следующий метод:

|  |  |
| --- | --- |
| 1  2  3  4 | public int Sum(int x, int y)  {      return x + y;  } |

У данного метода сигнатура будет выглядеть так: Sum(int, int)

И перегрузка метода как раз заключается в том, что методы имеют разную сигнатуру, в которой совпадает только название метода. То есть методы должны отличаться по:

* Количеству параметров
* Типу параметров
* Порядку параметров
* Модификаторам параметров

Например, пусть у нас есть следующий класс:

|  |
| --- |
| class Calculator  {      public void Add(int a, int b)      {          int result = a + b;          Console.WriteLine($"Result is {result}");      }      public void Add(int a, int b, int c)      {          int result = a + b + c;          Console.WriteLine($"Result is {result}");      }      public int Add(int a, int b, int c, int d)      {          int result = a + b + c + d;          Console.WriteLine($"Result is {result}");          return result;      }      public void Add(double a, double b)      {          double result = a + b;          Console.WriteLine($"Result is {result}");      }  } |

Здесь представлены четыре разных версии метода Add, то есть определены четыре перегрузки данного метода.

Первые три версии метода отличаются по количеству параметров. Четвертая версия совпадает с первой по количеству параметров, но отличается по их типу. При этом достаточно, чтобы хотя бы один параметр отличался по типу. Поэтому это тоже допустимая перегрузка метода Add.

То есть мы можем представить сигнатуры данных методов следующим образом:

|  |
| --- |
| Add(int, int)  Add(int, int, int)  Add(int, int, int, int)  Add(double, double) |

После определения перегруженных версий мы можем использовать их в программе:

|  |
| --- |
| Calculator calc = new Calculator();  calc.Add(1, 2); // 3  calc.Add(1, 2, 3); // 6  calc.Add(1, 2, 3, 4); // 10  calc.Add(1.4, 2.5); // 3.9 |

Также перегружаемые методы могут отличаться по используемым модификаторам. Например:

|  |
| --- |
| void Increment(ref int val)  {      val++;      Console.WriteLine(val);  }    void Increment(int val)  {      val++;      Console.WriteLine(val);  } |

В данном случае обе версии метода Increment имеют одинаковый набор параметров одинакового типа, однако в первом случае параметр имеет модификатор ref. Поэтому обе версии метода будут корректными перегрузками метода Increment.

А отличие методов по возвращаемому типу или по имени параметров не является основанием для перегрузки. Например, возьмем следующий набор методов:

|  |
| --- |
| int Sum(int x, int y)  {      return x + y;  }  int Sum(int number1, int number2)  {      return number1 + number2;  }  void Sum(int x, int y)  {      Console.WriteLine(x + y);  } |

Сигнатура у всех этих методов будет совпадать:

|  |
| --- |
| Sum(int, int) |

Поэтому данный набор методов не представляет корректные перегрузки метода Sum и **работать не будет.**

**Практическая часть.**

Построить приложение Windows Forms. Первичный вид окна представляет собой лишь поле для ввода размерности таблицы (таблиц) и кнопку построения шаблонов для ввода таблиц. В качестве шаблонов использовать:

а) массив TextBox-ов

b) компонент GridViev.

Предусмотреть заполнение матриц случайными числами с возможностью изменения элементов вручную.

Поле для ввода размерности таблицы (таблиц) и кнопка построения шаблонов должны быть заблокированы. После нажатия кнопки выполнения задания в окне не должно быть возможности для редактирования каких-либо компонентов, но должна присутствовать кнопка сброса, приводящая окно к первоначальному виду.

При работе с матрицами должны быть созданы отдельные методы для каждого действия. Для одного из методов построить перегрузку.

|  |  |  |
| --- | --- | --- |
| **Варианты** | **Задание 1** | **Задание 2** |
|  | **1** | **1** |
|  | **2** | **2** |
|  | **3** | **3** |
|  | **4** | **4** |
|  | **5** | **5** |
|  | **6** | **6** |
|  | **7** | **7** |
|  | **8** | **8** |
|  | **9** | **9** |
|  | **10** | **10** |
|  | **1** | **11** |
|  | **2** | **6** |
|  | **3** | **7** |
|  | **4** | **8** |
|  | **5** | **9** |
|  | **6** | **10** |

## Задание 1. Работа с двумерными массивами

1. Дана квадратная матрица А. Напишите процедуру вычисления ее определителя.
2. В целях экономии памяти треугольные матрицы хранятся в одномерных массивах размерности (n2 +n)/2. Напишите процедуры сложения, умножения и транспонирования таких матриц.
3. Матрица A является нижнетреугольной матрицей, а B – верхнетреугольной. Напишите процедуры сложения и умножения таких матриц, учитывающие их структуру.
4. Даны матрицы A и B одинаковой размерности. Вычислить их сумму C= A+B
5. Даны матрицы A и B. Вычислить их произведение C= A\*B, если это возможно.
6. Матрицы A и B являются нижнетреугольными матрицами одной размерности. Напишите процедуры сложения, умножения и транспонирования для таких матриц, учитывающие их структуру.
7. Матрицы A и B являются верхнетреугольными матрицами одной размерности. Напишите процедуры сложения, умножения и транспонирования для таких матриц, учитывающие их структуру.
8. Дана прямоугольная матрица A. Вычислить матрицу B = AT.
9. Дана квадратная матрица А. Напишите процедуру вычисления обратной матрицы А-1.
10. Дана квадратная матрица A. Построить транспонированную матрицу на том же месте.

***Задание 2* «Змейка»**

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по возрастанию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по возрастанию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по возрастанию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по возрастанию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по убыванию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по убыванию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по убыванию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по убыванию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по убыванию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по возрастанию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
| n |  |  |
|  |  |  |
|  |  | 1 |

1. Дана матрица размерности n\*n, заполненная случайными числами. Построить матрицу той же размерности, упорядочив элементы матрицы по убыванию и заполнив её змейкой по образцу.

|  |  |  |
| --- | --- | --- |
| 1 |  |  |
|  |  |  |
|  |  | n |

**Контрольные вопросы**

* + - 1. Что такое метод?
      2. Что такое сигнатура метода?
      3. Каким образом можно перегрузить метод?
      4. Параметры и аргументы метода
      5. Что представляют из себя возвращаемые значения метода
      6. Корректна ли следующая перегрузка методов? Если да, то почему? Если нет, то почему?

|  |
| --- |
| static void IncrementVal(ref int val)  {      val++;      Console.WriteLine(val);  }    static void IncrementVal(int val)  {      val++;      Console.WriteLine(val);  } |