

Основы программирования на C#

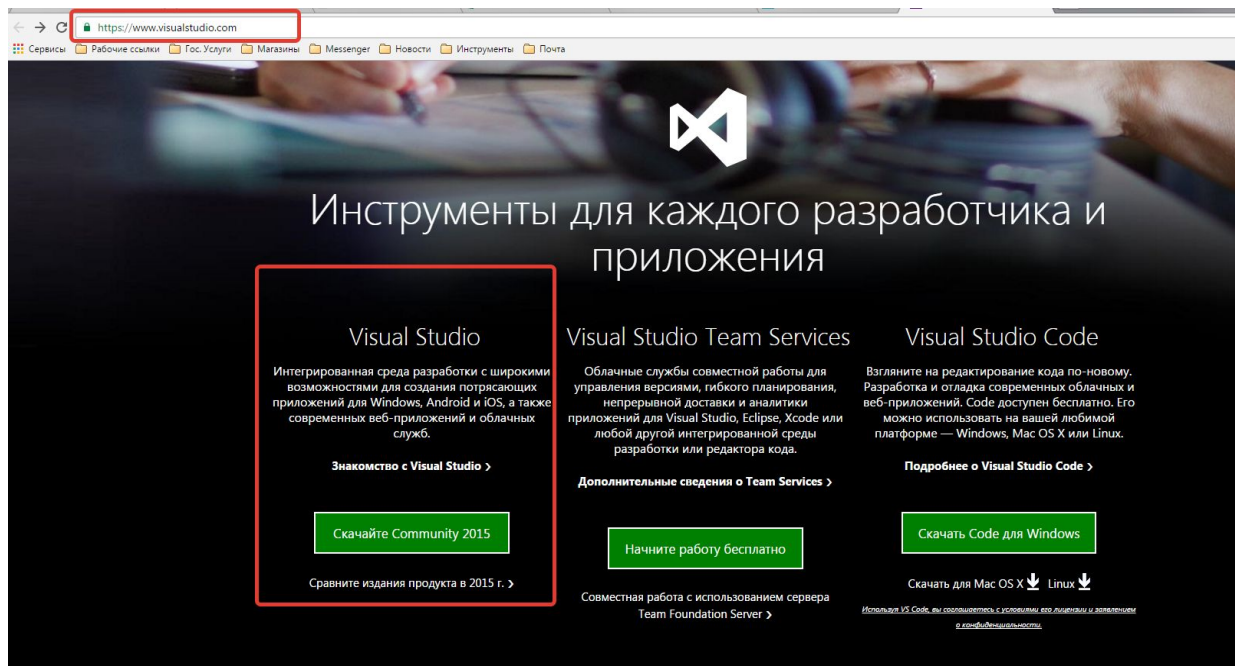
План занятия

- Установка и настройка Visual Studio
- Использование NuGet
- Создание простой программы
- Типы данных
- Концепции ввода/вывода
- Ключевые операции



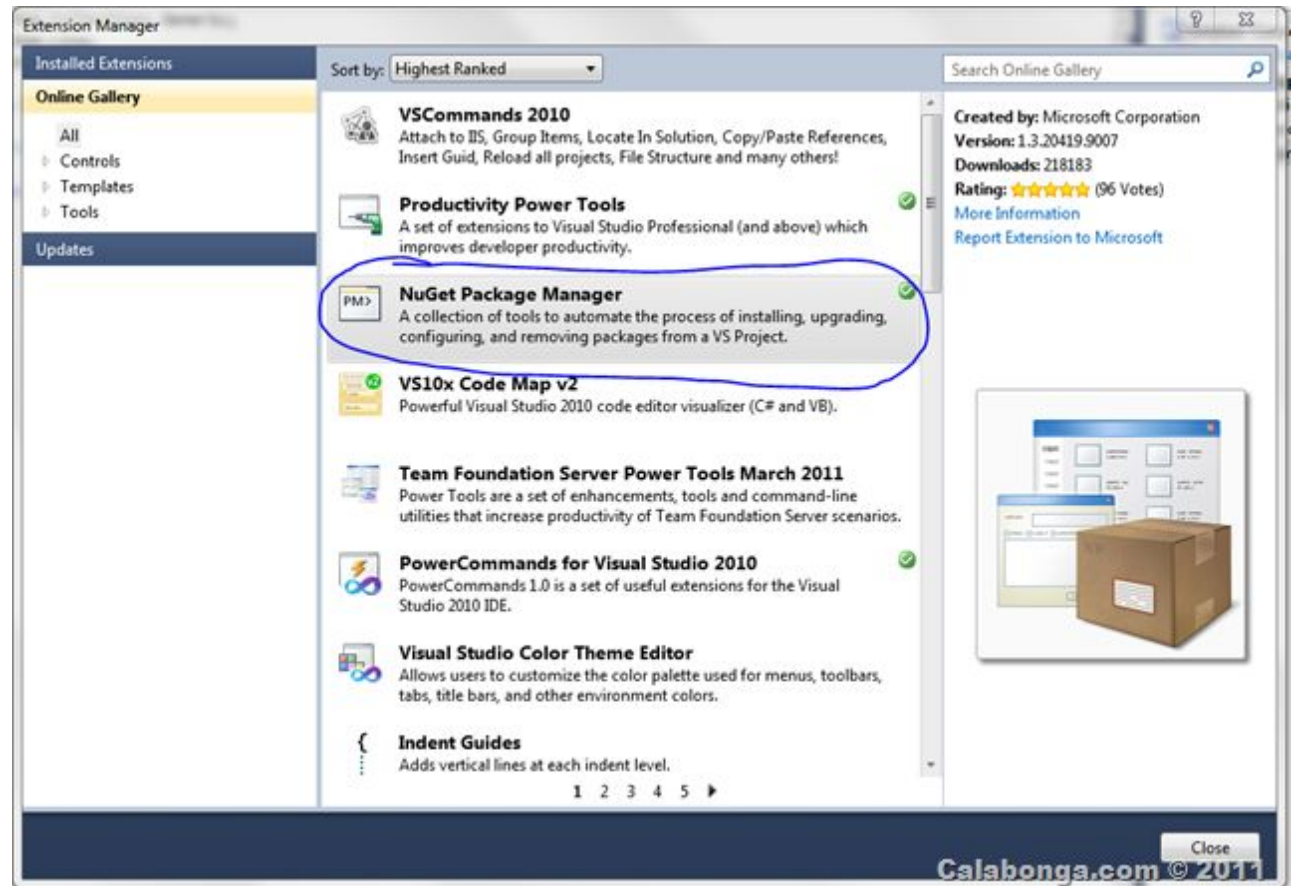
Установка и настройка Visual Studio

<https://www.visualstudio.com>



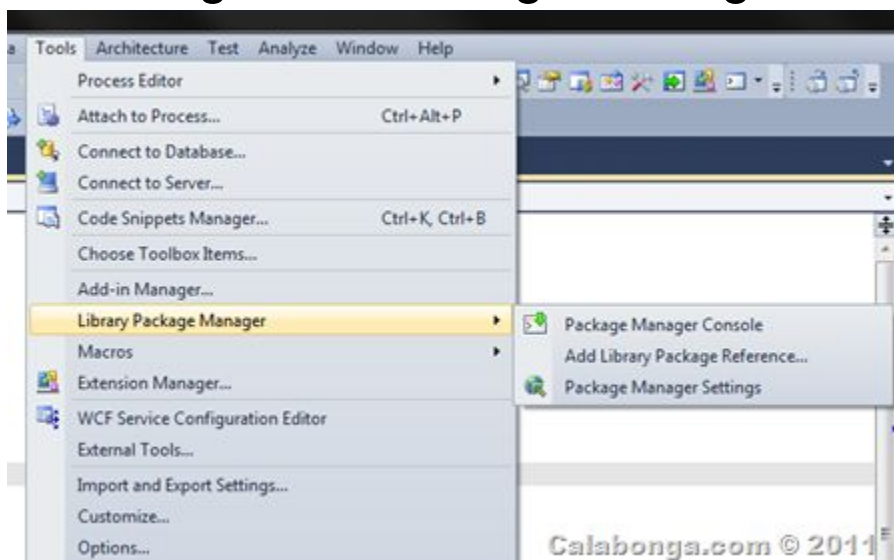
Использование NuGet – Управление библиотеками

NuGet – это замечательный инструмент, позволяющий разработчику легко управлять библиотеками в проектах любого типа.

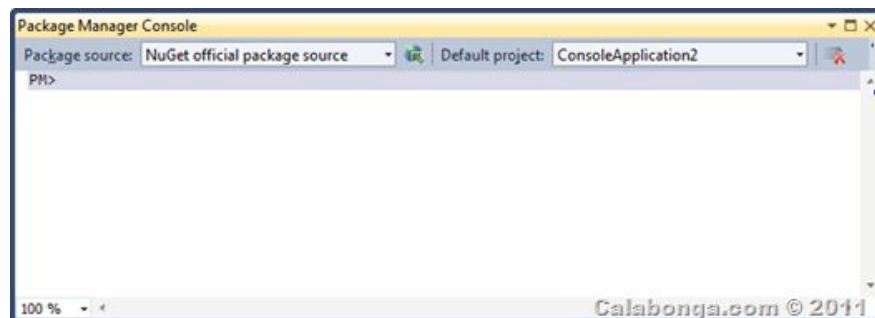


Использование NuGet – Управление библиотеками

После установки Nuget Package Manager в меню Visual Studio появились новые пункты, например, Tools → Library Package Manager → Package Manager Console.

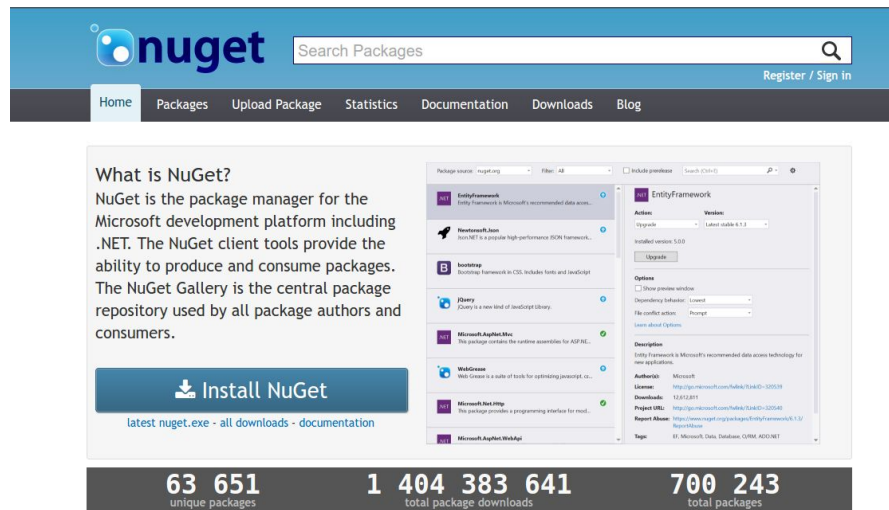


Открыв консоль можно писать команды для менеджера:



Использование NuGet – Управление библиотеками

<https://www.nuget.org/> - Вот тут можно искать интересующие нас библиотеки



Install-Package Selenium.WebDriver.ChromeDriver — нужно использовать в консоли студии)

Selenium.WebDriver.ChromeDriver 2.24.0

Install Chrome Driver(Win32) for Selenium WebDriver into your Unit Test Project.

"chromedriver.exe" is copied to bin folder from package folder when the build process.

NuGet package restoring ready, and no need to commit "chromedriver.exe" binary into source code control repository.

/ Selenium WebDriver用 Chrome Driver(Win32) を単体テストプロジェクトに追加します。

"chromedriver.exe" はビルド時にパッケージフォルダから bin フォルダへコピーされます。

NuGet パッケージの復元に対応済み、"chromedriver.exe" をソース管理リポジトリに登録する必要はありません。

/ The MSBuild script that contained this package is free and unencumbered software released into the public domain.

/ "chromedriver.exe" is licensed under the New BSD License.

To install Selenium.WebDriver.ChromeDriver, run the following command in the [Package Manager Console](#)

```
PM> Install-Package Selenium.WebDriver.ChromeDriver
```

Соглашения о написании кода на C#

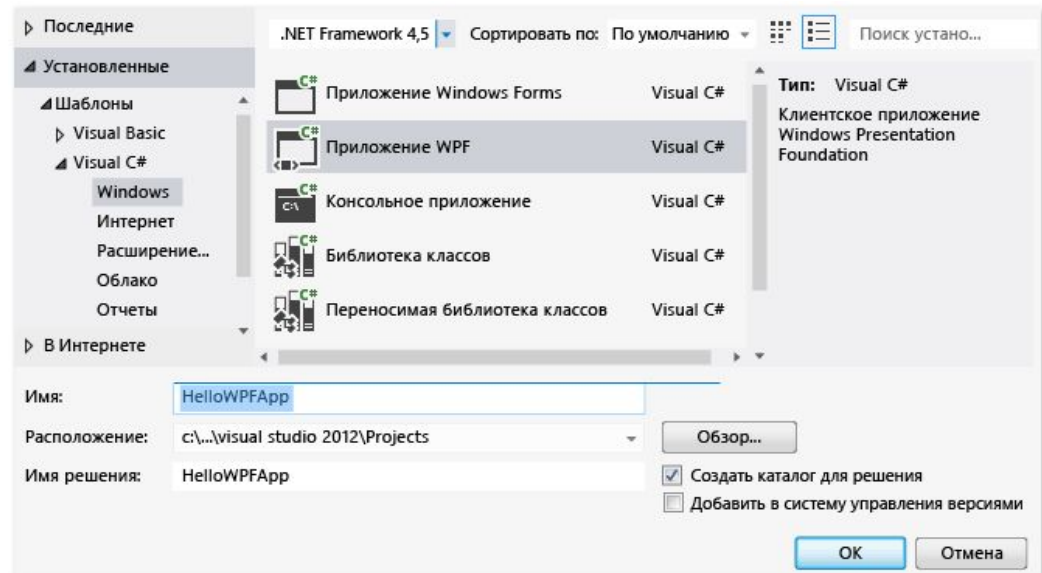
<https://msdn.microsoft.com> - много полезной информации.

- Соглашения о написании кода предназначены для реализации следующих целей:
- Создание согласованного вида кода, позволяющего читателям сосредоточиться на содержимом, а не на структуре.
 - Предоставление читателям возможности делать предположения, основанные на опыте, и поэтому быстрее понимать код.
 - Упрощение процессов копирования, изменения и обслуживания кода.
 - Предоставление лучших методик C#.

Создание простой программы в Visual Studio

Создание и запуск консольного приложения

- В меню Файл выберите Создать, Проект.
- Откроется диалоговое окно Новый проект.
- Разверните узел Установленные, разверните Шаблоны, разверните Visual C#, а затем выберите Консольное приложение.
- В поле Имя введите имя для проекта и нажмите кнопку ОК.
- ...
- Нажмите клавишу F5 для запуска проекта.



Виды типов объектов

Value type

- Живут в стеке
- Гарантированно будет живым как минимум до тех пор, пока текущая «точка исполнения» находится в области видимости локальной переменной

Reference type

- Живут в динамической памяти
- Уничтожаются сборщиком мусора при отсутствии ссылок на них

Простые value type

- **Целые**

byte, short, **int**, long,
sbyte, ushort, uint, ulong

- **Вещественные**

float, double

- **Финансовый** (повышенной точности)

decimal

- **Логический**

bool

- **Символьный**

char

<http://msdn.microsoft.com/ru-ru/library/ya5y69ds.aspx>

Value type, определяемые пользователем

Структуры

```
struct MyPoint
{
    public int x { get; set; }
    public int y { get; set; }
}
```

```
MyPoint p = new MyPoint();
p.x = 5;
p.y = 6;
```

Перечисления

```
enum DaysOfWeek
{
    Sunday,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
}
```

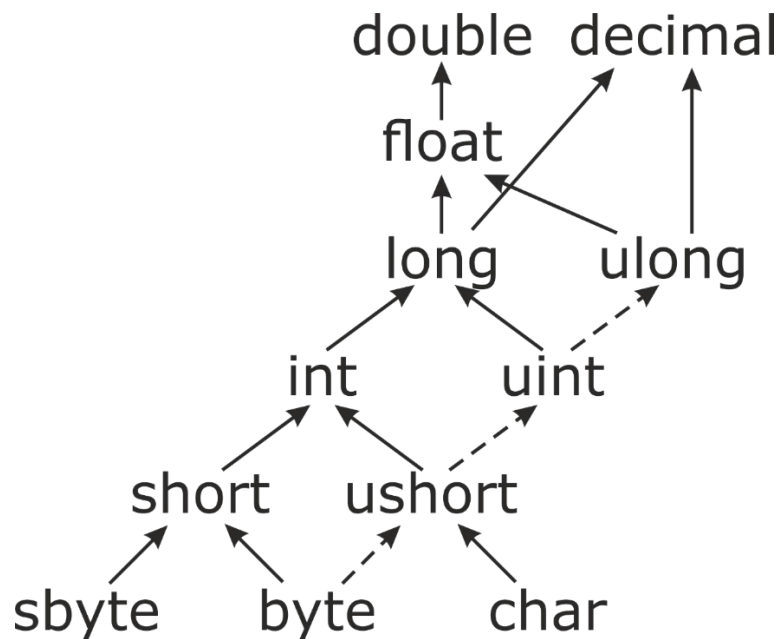
```
DaysOfWeek day =
    DaysOfWeek.Monday;
```

<http://msdn.microsoft.com/ru-ru/library/0taef578.aspx>

[http://msdn.microsoft.com/ru-ru/library/sbvt4032\(v=vs.90\).aspx](http://msdn.microsoft.com/ru-ru/library/sbvt4032(v=vs.90).aspx)

Приведение value type

Неявное приведение базовых типов C#



—————> предпочтительный
 путь
- . - . - > допустимый
 путь

Явное приведение

```
int x = 2, y = 5;  
double z = ((double)x) / y;
```

Операции

- Арифметические

+ - / * %

- Сравнения

> < >= <= == !=

- Логические

&& || ! & |

- Битовые

& | ~ ^

- Присваивания

= += -= /= *= %= ++ -- &= |= ...

Операторы языка C#

•Условные

if, switch, ?:, ??

•Циклы

while, do/while, for, foreach

•Перехода

break, continue, goto, return

Оператор If

```
if(условие)
{
    действия, если условие
    истинно
}
else
{
    действия, если условие
    ложно
}
```

```
if (a > b)
{
    max = a;
}
else
{
    max = b;
}
```

Конструкция else if

```
if(условие1)
{
    действия
}
else if (условие2)
{
    действия
}
else if (условие3)
{
    действия
}
```

Оператор switch

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

Оператор switch

```
string str = Console.ReadLine();
int n;
switch (str)
{
    case "Понедельник": n = 5; break;
    case "Вторник":      n = 4; break;
    case "Среда":        n = 3; break;
    case "Четверг":      n = 2; break;
    case "Пятница":      n = 1; break;
    default:             n = 0; break;
}
Console.WriteLine("Осталось работать {0} дней", n);
```

Оператор for

```
for([<инициализация>]; [<выражение>]; [<модификация>])  
{  
    тело цикла  
}
```

```
Console.Write("N =");  
string str = Console.ReadLine();  
int n = int.Parse(str);  
for (int i = 1; i <= n; i++)  
{  
    Console.Write(" {0}", i);  
}  
Console.ReadLine();
```

Оператор while

while (условие)

{

тело цикла

}

```
Console.Write("N =");  
string str = Console.ReadLine();  
int n = int.Parse(str);  
int i = 1;  
while (i <= n)  
{  
    Console.Write(" {0}", i++);  
}  
Console.ReadLine();
```


Оператор do while

```
do
{
    тело цикла
}while(условие)
```

```
Console.Write("N =");
string str = Console.ReadLine();
int n = int.Parse(str);
int i = 1;
do
{
    Console.Write(" {0}", i++);
} while (i <= n);
Console.ReadLine();
```

Вывод данных

Отдельные значения

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

Ручное объединение строк

```
Console.WriteLine("x = " + x + ", y = " + y);
```

Форматировани

е

```
Console.WriteLine("x = {0}, y = {1}", x, y);
```

Ввод данных

- Ввод данных с клавиатуры возможен только в строковом формате;
- Для приведения строки к конкретному типу, как правило, используются методы `Parse` и `TryParse`;
- При ошибке преобразования типа может произойти исключительная ситуация, однако предусматривать её не обязательно.

Пример ввода данных

```
Console.Write("N = ");  
  
string str = Console.ReadLine();  
int n = int.Parse(str);  
  
Console.WriteLine("Your input: {0}", n);  
Console.ReadLine();
```

Блок try ... catch ... finally

Синтаксис:

```
try
{
    действия_способные_привести
    к_исключительной_ситуации
}
catch [(тип_исключения [переменная_исключения])]
{
    обработка_исключения
}
[finally
{
    действия_выполняемые_вне_зависимости
    от_результата_блока_try
}]
```

Обработка неверного ввода перехватом исключения

```
string str = Console.ReadLine();

try
{
    int n = int.Parse(str);
}
catch (FormatException)
{
    Console.WriteLine("Wrong input!");
    return;
}
```


Одномерные массивы

- Объявление массива
`тип_элементов[] имя;`
- Создание массива
`имя = new тип_элементов
[число_элементов];`
- Доступ к элементам
`имя[номер_элемента]`

0	1	2	3	4	5	6	7
5	7	8	0	0	0	0	0

```
int[] arr;  
arr = new int[8];  
arr[0] = 5;  
arr[1] = 7;  
arr[2] = arr[1] + 1;
```

Одномерные массивы: сокращённая инициализация

- Стандартная инициализация

```
int[] arr = new int[2];  
arr[0] = 0;  
arr[1] = 1;
```

- Задание значений

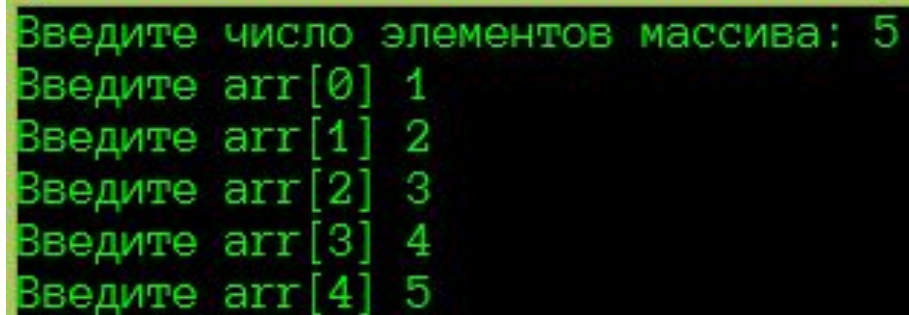
```
int[] arr = new int[2] {0, 1};  
int[] arr = new int[] {0, 1};
```

- Автоматическое определение числа элементов

```
int[] arr = {0, 1};
```

Ввод массива с клавиатуры

```
class Program
{
    static void Main(string[] args)
    {
        int n;
        Console.Write("Введите число элементов массива: ");
        n = int.Parse(Console.ReadLine());
        int[] arr = new int[n];
        for (int i = 0; i < arr.Length; i++)
        {
            Console.Write("Введите arr[{0}] ", i);
            arr[i] = int.Parse(Console.ReadLine());
        }
        Console.ReadLine();
    }
}
```

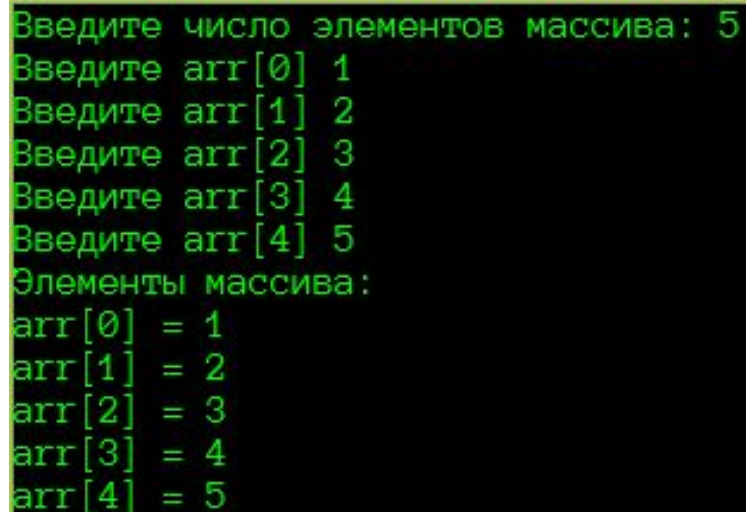


```
Введите число элементов массива: 5
Введите arr[0] 1
Введите arr[1] 2
Введите arr[2] 3
Введите arr[3] 4
Введите arr[4] 5
```

Вывод массива на экран

```
static void Main(string[] args)
{
    Ввод массива

    Console.WriteLine("Элементы массива:");
    for (int i = 0; i < n; i++)
    {
        Console.WriteLine("arr[{0}] = {1}",
            i, arr[i]);
    }
    Console.ReadLine();
}
```



Введите число элементов массива: 5
Введите arr[0] 1
Введите arr[1] 2
Введите arr[2] 3
Введите arr[3] 4
Введите arr[4] 5
Элементы массива:
arr[0] = 1
arr[1] = 2
arr[2] = 3
arr[3] = 4
arr[4] = 5

Цикл foreach

- Синтаксис

```
foreach (тип_элемента имя in набор)
{
    тело_цикла
}
```

- Пример

```
foreach (int x in arr)
{
    Console.WriteLine(x);
}
```

Многомерные массивы

- Объявление массива
`тип_элементов[,] имя;`
- Создание массива
`имя = new тип_элементов [N, K];`
- Доступ к элементам
`имя[n, k]`

	0	1	2	3	4	5
0	5	7	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	8	0	0	0	0

```
int[,] arr;  
arr = new int[4,6];  
arr[0,0] = 5;  
arr[0,1] = 7;  
arr[3,1] = arr[0,1] + 1;
```


Спасибо за внимание!