Блок 1. Базовый C#

1.2 — Основные конструкции

План занятия

- Основные конструкции
- Массивы
- Методы
- Способы передачи аргументов

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;
}
```

Оператор switch

```
switch(выражение)
{
      case значение_1:
         действия, если выражение равно значение_1;
            оператор перехода;
      case значение_2:
         действия, если выражение равно значение_2;
            оператор перехода;
...

      default:
         действия, если выражение не совпало ни с
значений;
            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 switch

```
Actions n;
// ...
switch (n)
{
    case Actions.Read:
    case Actions.ReadWrite:
        int i = 10;
        break;

    case Actions.Write:
        i = 4;
        goto case Actions.Append;

    case Actions.Append:
        i = 3;
        break;
}
```

Конструкция else if

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

Оператор ?

условие
  ? значение_если_истина
  : значение_если_ложь

```
int max = x > y ? x : y;
```

```
int max;
if (x > y)
{
    max = x;
}
else
{
    max = y;
}
```

```
int a = (x > (y != 0 ? 1 : 0) ? y : 1) != 0 ?
    (y - 2 > 0) ? y - 2 : -(y - 2) : 0;
```

Оператор ??

Значение1 ?? Значение2, если значение1 = null

```
object max = x ?? y;
```

```
object max = (x != null) ? x : y;
```

```
object max;
if (x != null)
{
    max = x;
}
else
{
    max = y;
}
```

Оператор ?? не может применяться к объектам типов, не поддерживающих значение **null**

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();
```

Каждый из блоков for(;;) является необязательным и может быть опущен

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();
```

Выбор типа цикла

- Вывести на экран числа от 1 до 100.
- Применить действие ко всем элементам массива.
- Вывести на экран первые 100 строк файла.
- Подсчитать сумму бесконечного ряда с заданной точностью.

Одномерные массивы

- Объявление массива

  `тип_элементов[] имя;`

- Создание массива

  `имя = new тип_элементов [число_элементов];`

- Доступ к элементам

  `имя[номер_элемента]`

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

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

Одномерные массивы: сокращённая инициализация

- Стандартная инициализация

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

- Задание значений

```
int[] arr1 = new int[2] { 0, 1 };
int[] arr2 = new int[] { 0, 1 };
```

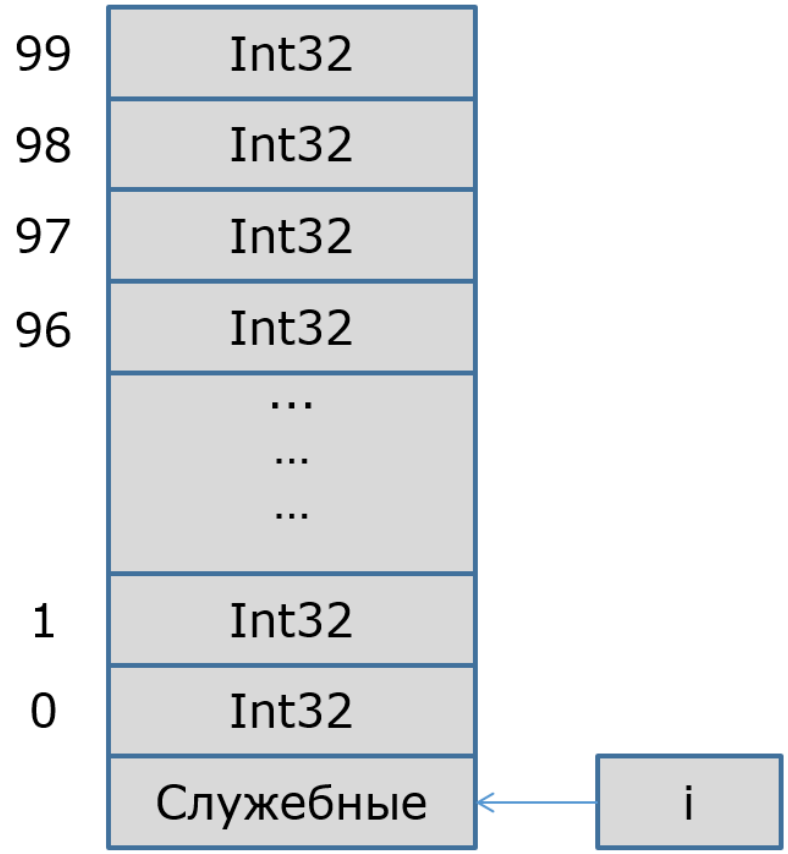
- Автоматическое определение числа элементов

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

Элементы массивов

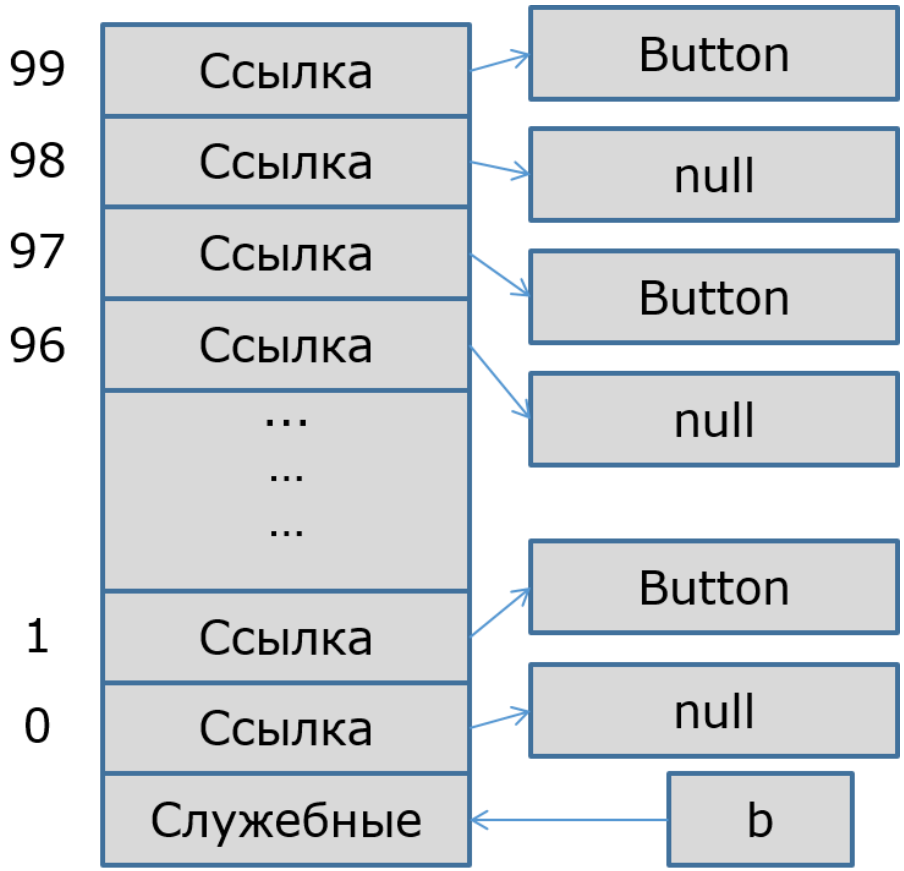
Структуры

```
int[] i = new int[100];
```



Классы

```
Button[] b = new Button[100];
```



Ввод массива с клавиатуры

```
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

```
// Ввод элементов

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
```

Заполнение массива случайными числами

```
int n;
Console.Write("Введите число элементов массива: ");
n = int.Parse(Console.ReadLine());
int[] arr = new int[n];

Random r = new Random();
for (int i = 0; i < arr.Length; i++)
{
    arr[i] = r.Next(100);
}

// Вывод элементов
```

```
Введите число элементов массива: 5
Элементы массива:
arr[0] = 8
arr[1] = 80
arr[2] = 17
arr[3] = 57
arr[4] = 69
```

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]
```

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

	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

Рваные (jagged) массивы

- Объявление массива

  тип_элементов[][] имя;

- Создание массива ссылок

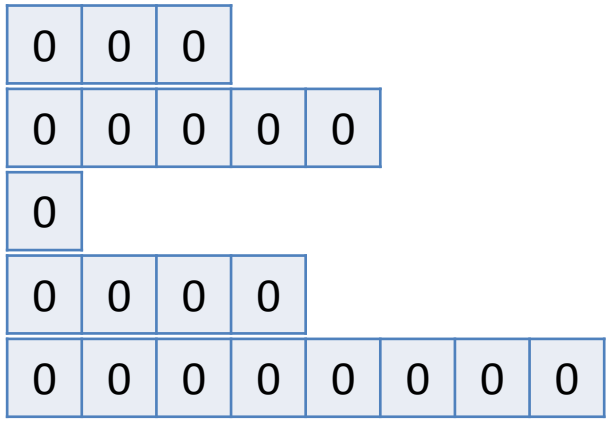
  имя = new тип_элементов [N][];

- Создание массивов

  имя[i] = new тип_элементов [K];

- Доступ к элементам

  имя[n][k]




```
int[][] arr = new int[5][];
arr[0] = new int[3];
arr[1] = new int[5];
arr[2] = new int[1];
arr[3] = new int[4];
arr[4] = new int[8];
```

Array

- Абстрактный класс, от которого неявно наследуются все массивы
- Свойства
  - Length
  - LongLength
  - Rank
- Методы
  - GetLength(размерность)
  - GetLowerBound(размерность)
  - GetUpperBound(размерность)

Статические методы класса Array

- CopyTo
- IndexOf
- LastIndexOf
- Reverse
- Sort
- Exists
- Find
- FindAll

```
int[] arr = { 1, -2, 3, 4, -5, 2, 7, 1, 9 };
int res1 = Array.IndexOf(arr, 1);
int res2 = Array.LastIndexOf(arr, 1);
Array.Sort(arr);
int[] res = Array.FindAll(arr, x => x < 0);
```

Объявление методов

- Методы — функции, явно закреплённые за какими-либо типами данных (статические методы) или за их объектами (экземплярные, инстансные, нестатические методы);
- Метод должен либо возвращать одно значение конкретного типа, либо ничего не возвращать (вместо типа данных указывается ключевое слово **void**);
- Синтаксис:

```
[спецификаторы] тип_результата
имя_метода(список_аргументов)
{
  тело_метода
  [return результат]
}
```

Пример 1

```
static bool IsEven(int number)
{
    return number % 2 == 0;
}

static void Main(string[] args)
{
    Console.WriteLine("Введите целое число");
    int num = int.Parse(Console.ReadLine());
    if (IsEven(num))
    {
        Console.WriteLine("Введённое число является чётным");
    }
    else
    {
        Console.WriteLine("Введённое число является нечётным");
    }
    Console.Read();
}
```

Пример 2

```
static void ShowArray(int[] arr)
{
    foreach (int elem in arr)
    {
        Console.Write("\t{0}", elem);
    }
    Console.WriteLine();
}

static void Main(string[] args)
{
    int[] arr = { 5, 22, 2, 4, 6 };
    Console.WriteLine("Исходный массив");
    ShowArray(arr);
    Console.WriteLine("Сортированный массив");
    Array.Sort(arr);
    ShowArray(arr);
    Console.Read();
}
```

Именованные и необязательные аргументы

- Именованные аргументы позволяют при вызове метода указать значение аргумента в зависимости от его имени, а не позиции в списке аргументов;
- Значения для необязательных аргументов можно опускать — им будет присвоено указанное в сигнатуре значение по умолчанию.

Пример 1

```
User CreateUser(string login, string password, bool isAdmin = false)
{
    // ...
}

void CreateUsers()
{
    CreateUser("Иванов", "123");
    CreateUser(login: "Иванов", password: "123");
}
```

CIL:

```
private void CreateUsers()
{
        Program.CreateUser("Иванов", "123", false);
        Program.CreateUser("Иванов", "123", false);
}
```

Пример 2

```
public static int Sum(int op1, int op2, int op3 = 0, int op4 = 0)
{
    return op1 + op2 + op3 + op4;
}

static void Main(string[] args)
{
    Console.WriteLine(Sum(1, 2, 3, 4));
    Console.WriteLine(Sum(1, 2));
    Console.WriteLine(Sum(1, 2, 3));
}
```

CIL:

```
private static void Main(string[] args)
{
    Console.WriteLine(Program.Sum(1, 2, 3, 4));
    Console.WriteLine(Program.Sum(1, 2, 0, 0));
    Console.WriteLine(Program.Sum(1, 2, 3, 0));
}
```

Пример 3

```
public static double F(double m1, double m2 = 5.97e24,
    double r = 6378000)
{
    return 6.67e-11 * m1 * m2 / (r * r);
}

static void Main(string[] args)
{
    Console.WriteLine(F(1));
    Console.WriteLine(F(m1: 1, r: 6388000));
    Console.WriteLine(F(1, r: 6388000));
    Console.WriteLine(F(r: 10, m1: 10, m2: 10));
}
```

CIL:

```
private static void Main(string[] args)
{
    Console.WriteLine(Program.F(1.0, 5.97E+24, 6378000.0));
    Console.WriteLine(Program.F(1.0, 5.97E+24, 6388000.0));
    Console.WriteLine(Program.F(1.0, 5.97E+24, 6388000.0));
    double r = 10.0;
    Console.WriteLine(Program.F(10.0, 10.0, r));
}
```

Способы передачи параметров

- По значению (используется по умолчанию)
- По ссылке (модификатор ref)
- По ссылке для возврата (модификатор out)
- Как массив (модификатор params)

Передача параметров по значению

- В метод передаётся копия стекового значения параметра (формальный параметр):
  - Для структур создаётся новая копия объекта;
  - Для классов создаётся новая копия указателя на тот же самый объект;
- Любые действия, совершаемые с формальным параметром, не ведут к изменению фактического параметра;
- В качестве фактического параметра могут выступать выражения (при передаче по значению).

Пример

```
static void Funct(int par)
{
    Console.WriteLine(par);
    par = 7;
    Console.WriteLine(par);
}

static void Main(string[] args)
{
    int x = 5;
    Console.WriteLine(x);
    Funct(x);
    Console.WriteLine(x);
    Console.Read();
}
```

Передача параметров по ссылке

- В метод передаётся ссылка на фактический параметр:
  - Для структур создаётся ссылка на оригинальный объект;
  - Для классов создаётся ссылка на оригинальный указатель на объект;
- Для обозначения способа передачи используется модификатор аргумента **ref**;
- Параметром может выступать только инициализированная переменная или поле объекта (что-то, имеющее адрес);

Передача массива в метод

- Массивы являются объектами классов вне зависимости от типа их элементов;
- При передаче массива по значению метод может напрямую работать с его элементами;
- При передаче массива по ссылке метод может повторно инициализировать массив.

Пример

```
static void Funct(ref int par)
{
    Console.WriteLine(par);
    par = 7;
    Console.WriteLine(par);
}

static void Main(string[] args)
{
    int x = 5;
    Console.WriteLine(x);
    Funct(ref x);
    Console.WriteLine(x);
    Console.Read();
}
```

Передача параметров по ссылке для возврата

- В метод передаётся ссылка на фактический параметр;
- Для обозначения способа передачи используется модификатор аргумента **out**;
- Параметром может выступать только переменная или поле объекта;
- Переменную можно не инициализировать перед передачей в метод по **out**;
- Формальный параметр обязан быть инициализирован в методе.

Пример

```
static void Funct(out int n)
{
    n = int.Parse(Console.ReadLine());
    Console.WriteLine(n);
}

static void Main(string[] args)
{
    int x;
    Funct(out x);
    Console.WriteLine(x);
    Console.Read();
}
```

Передача неопределённого числа параметров

- Набор аргументов принимается и обрабатывается методом как переданный по значению массив;
- Для обозначения способа передачи используется модификатор аргумента **params**;
- Метод может принимать не более одного такого параметра;
- Параметр с модификатором params должен быть последним в списке параметров;
- Допускается явная передача в метод готового массива нужной размерности и типа.

Пример

```
static int Max(params int[] par)
{
    int max = par[0];
    for (int i = 1; i < par.Length; i++)
    {
        if (max < par[i])
        {
            max = par[i];
        }
    }
    return max;
}

static void Main(string[] args)
{
    int max = Max(3, 7, 4);
    Console.WriteLine(max);
    Console.Read();
}
```

Пример

```
22          class Point
23          {
24              public int x;
25              public int y;
26          }
27
28          static void Funct(Point p)
29          {
30              p.x = 5;
31              p = new Point { x = 10, y = 10 };
32          }
33
34          static void Main(string[] args)
35          {
36              Point poi = new Point { x = 1, y = 1 };
37              Funct(poi);
38              Console.WriteLine("({0}, {1})", poi.x, poi.y);
39              Console.Read();
40          }
```

Критерии создания методов

- Соответствие принципу единственной ответственности (1 метод — 1 действие);
- Отсутствие повторяющихся фрагментов кода (т.н. copy-paste);
- Разделение бизнес-логики и логики представления;
- Размер метода не должен превышать 1–2 экрана кода.

Спасибо за внимание!

Контактная информация:
Дмитрий Вересkун
Инструктор
EPAM Systems, Inc.
Адрес: Саратов, Рахова, 181
Email: Dmitry_Vereskun@epam.com
http://www.epam.com