Оглавление

[Литералы 2](#_Toc161493395)

[Типы данных 2](#_Toc161493396)

[Консольный ввод-вывод 2](#_Toc161493397)

[Логические операции 3](#_Toc161493398)

[Операции присваивания 3](#_Toc161493399)

[Массивы 3](#_Toc161493400)

[Одномерные массивы 3](#_Toc161493401)

[Многомерные массивы 3](#_Toc161493402)

[Массив массивов 4](#_Toc161493403)

[Методы 4](#_Toc161493404)

[Передача параметров по ссылке. Выходные параметры 4](#_Toc161493405)

[Модификатор ref 4](#_Toc161493406)

[Модификатор out 5](#_Toc161493407)

[Модификатор in 5](#_Toc161493408)

[Модификатор ref readonly (c C#12) 5](#_Toc161493409)

[Массив параметров и ключевое слово params 5](#_Toc161493410)

[Локальные функции 6](#_Toc161493411)

[Конструкция switch 6](#_Toc161493412)

[Перечисления enum 6](#_Toc161493413)

# Литералы

0b11 **-** 3 число в двоичной форме

0x0A **-** 10 число в шестнадцатеричной форме

3.2e3 **(**3.2E3**)** **-** 3.2 **\*** 10**^**3

'\x78' **-** x символ в ASCII

'\u0420' **-** P символ в Unicode

# Типы данных

bool**:** хранит значение **true** или **false** **(**логические литералы**).**

byte**:** хранит целое число от 0 до 255 и занимает 1 байт**.**

sbyte**:** хранит целое число от **-**128 до 127 и занимает 1 байт**.**

short**:** хранит целое число от **-**32768 до 32767 и занимает 2 байта**.**

ushort**:** хранит целое число от 0 до 65535 и занимает 2 байта**.**

int**:** хранит целое число от **-**2147483648 до 2147483647 и занимает 4 байта**.**

uint**:** хранит целое число от 0 до 4294967295 и занимает 4 байта**.** Суффикс u **(**U**)** **-** 10U

long**:** хранит целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт**.** Суффикс l **(**L**)** **-** 10L

ulong**:** хранит целое число от 0 до 18 446 744 073 709 551 615 и занимает 8 байт**.** Суффикс ul **(**UL**)** **-** 10L

float**:** хранит число с плавающей точкой от **-**3.4e38 до 3.4e38 и занимает 4 байта**.** Суффикс f **(**F**)** **-** 3.14F

double**:** хранит число с плавающей точкой от ±5.0e-324 до ±1.7e308 и занимает 8 байта**.**

decimal**:** хранит десятичное дробное число**.** Если употребляется без десятичной запятой**,** имеет значение от ±1.0**\***10**-**28 до ±7.9228**\***1028**,** может хранить 28 знаков после запятой и занимает 16 байт**.** Суффикс m **(**M**)** **-** 3.14M

char**:** хранит одиночный символ в кодировке Unicode и занимает 2 байта**.**

string**:** хранит набор символов Unicode**.**

**object:** может хранить значение любого типа данных и занимает 4 байта на 32**-**разрядной платформе и 8 байт на 64**-**разрядной платформе**.**

# Консольный ввод-вывод

Convert**.**ToInt32**()** **(**преобразует к типу int**)**

Convert**.**ToDouble**()** **(**преобразует к типу double**)**

Convert**.**ToDecimal**()** **(**преобразует к типу decimal**)**

Console**.**Write**(**"Hello"**);** // Hello

int age **=** Convert**.**ToInt32**(**Console**.**ReadLine**());**

double height **=** Convert**.**ToDouble**(**Console**.**ReadLine**());**

Console**.**WriteLine**(**$"Age = {age} years old, height = {height} meters"**);**

Console**.**WriteLine**(**"Age = {0} years old, height = {1} meters"**,** age**,** height**);**

# Логические операции

int x **=** 2**;** // 010

int y **=** 5**;** //101

x**&**y **-** 0 // 000 AND

x**|**y **-** 7 // 111 OR

x**^**y **-** 7 // 111 XOR

**~**x **-** **-**2 // 1...010 старший бит равен 1

x**<<**1 **-** 4 // 100

x**>>**1 **-** 1 // 001

# Операции присваивания

int a**,** b**,** c**;**

a **=** b **=** c **=** 34**;**

**+=**

**-=**

**\*=**

**/=**

**%=**

**&=**

**|=**

**^=**

**>>=**

**<<=**

# Массивы

## Одномерные массивы

int**[]** nums **=** **new** int**[**4**];**

int**[]** nums **=** **new** int**[**4**]** **{** 1**,** 2**,** 3**,** 5 **};**

int**[]** nums **=** **new** int**[]** **{** 1**,** 2**,** 3**,** 5 **};**

int**[]** nums **=** **new[]** **{** 1**,** 2**,** 3**,** 5 **};**

int**[]** nums **=** **{** 1**,** 2**,** 3**,** 5 **};**

c C#12

int**[]** nums **=** **[** 1**,** 2**,** 3**,** 5 **];**

int**[]** nums **=** **[];** // пустой массив

nums**[**0**]** **-** 1

nums**[^**1**]** **-** 5m

**for(**int i **=** 0**;** i **<** nums**.**Length**;** i**++)**

**{**

Console**.**WriteLine**(**nums**[**i**]);**

**}**

**foreach(**var n **in** nums**)**

Console**.**WriteLine**(**n**);**

## Многомерные массивы

int**[,]** nums**;**

int**[,]** nums **=** **new** int**[**2**,** 3**];**

int**[,]** nums **=** **new** int**[**2**,** 3**]** **{** **{** 0**,** 1**,** 2 **},** **{** 3**,** 4**,** 5 **}** **};**

int**[,]** nums **=** **new** int**[,]** **{** **{** 0**,** 1**,** 2 **},** **{** 3**,** 4**,** 5 **}** **};**

int**[,]** nums **=** **new** **[,]{** **{** 0**,** 1**,** 2 **},** **{** 3**,** 4**,** 5 **}** **};**

int**[,]** nums **=** **{** **{** 0**,** 1**,** 2 **},** **{** 3**,** 4**,** 5 **}** **};**

**foreach** **(**int i **in** numbers**)**

Console**.**Write**(**$"{i} "**);** // 012345

**for** **(**val i **=** 0**;** i **<=** nums**.**GetUpperBound**(**0**);** i**++)**

**for** **(**val j **=** 0**;** j **<=** nums**.**GetUpperBound**(**1**);** j**++)**

Console**.**Write**(**nums**[**i**,**j**]);** // 012345

## Массив массивов

int**[][]** nums **=** **new** int**[**3**][];**

nums**[**0**]** **=** **new** int**[**2**]** **{** 1**,** 2 **};** // выделяем память для первого подмассива

nums**[**1**]** **=** **new** int**[**3**]** **{** 1**,** 2**,** 3 **};** // выделяем память для второго подмассива

nums**[**2**]** **=** **new** int**[**5**]** **{** 1**,** 2**,** 3**,** 4**,** 5 **};** // выделяем память для третьего подмассива

int**[][]** nums **=** **{**

**new** int**[]** **{** 1**,** 2 **},**

**new** int**[]** **{** 1**,** 2**,** 3 **},**

**new** int**[]** **{** 1**,** 2**,** 3**,** 4**,** 5 **}**

**};**

**foreach** **(**int**[]** row **in** nums**)**

**foreach** **(**int num **in** row**)**

Console**.**Write**(**num**);** // 12345

**for** **(**val i **=** 0**;** i **<** nums**.**Length**;** i**++)**

**for** **(**val j **=** 0**;** j **<** nums**[**i**].**Length**;** j**++)**

Console**.**Write**(**nums**[**i**][**j**]);** \\ 12345

# Методы

int Sum**(**int x**,** int y **=** 0**)**

**{**

**return** x **+** y**;**

**}**

int Sum**(**int x**,** int y **=** 0**)** **=>** x **+** y**;**

Sum**(**4**)** **-** 4

Sum**(**4**,** 5**)** **-** 9

Sum**(**y**:**10**,** x**:**5**)** **-** 15

# Передача параметров по ссылке. Выходные параметры

## Модификатор ref

При передаче значений параметрам по ссылке метод получает адрес переменной в памяти**.**

Параметр с модификатором **ref** обязан быть инициализирован перед передачей в метод**.**

void Increment**(ref** int n**)**

**{**

n**++;**

**}**

int number **=** 5**;**

Increment**(ref** number**)** // number = 6

## Модификатор out

Методы**,** использующие параметры **out,** обязательно должны присваивать им определенное значение**,** даже если оно им уже присвоено где**-**то в коде перед вызовом метода **(**в отличие от **ref).**

void Sum**(**int x**,** int y**,** **out** int sum**,** **out** int mul**)**

**{**

sum **=** x **+** y**;**

mul **=** x **\*** y**;**

**}**

int numSum**;**

Sum**(**10**,** 5**,** **out** numSum**,** **out** int numMul**);** // numSum = 15, numMul = 50 (переменную можно определить непосредственно при вызове метода)

## Модификатор in

Модификатор **in** указывает**,** что данный параметр будет передаваться в метод по ссылке**,** однако внутри метода его значение параметра нельзя будет изменить**.**

int Sum**(in** int x**,** **in** int y **=** 0**)** **=>** x **+** y**;**

## Модификатор ref readonly (c C#12)

**ref-**параметры только для чтения**.**

void Increment**(ref** **readonly** int n**)**

**{**

// n++; // нельзя, иначе будет ошибка компиляции

**}**

int number **=** 5**;**

Increment**(ref** number**);**

# Массив параметров и ключевое слово params

int Sum**(params** int**[]** numbers**)**

**{**

int result **=** 0**;**

**foreach** **(**var n **in** numbers**)**

result **+=** n**;**

**return** result**;**

**}**

int**[]** nums **=** **{** 1**,** 2**,** 3**,** 4**,** 5**};**

Sum**(**nums**);**

Sum**(**1**,** 2**,** 3**,** 4**);**

Sum**();**

# Локальные функции

Локальные функции представляют функции**,** определенные внутри других методов**.**

Локальная функция**,** как правило**,** содержит действия**,** которые применяются только в рамках ее метода**.**

bool AreEqual**(**int**[]** nums1**,** int**[]** nums2**)**

**{**

int sum1 **=** Sum**(**nums1**);**

int sum2 **=** Sum**(**nums2**);**

**return** sum1 **==** sum2**;**

int Sum**(**int**[]** nums**)** **=>** nums**.**Sum**();**

// static int Sum(int[] nums) => nums.Sum();

// static функции не могут обращаться к переменным окружения, то есть метода,

// в котором статическая функция определена в отличие от нестатических функций.

**}**

# Конструкция switch

**switch** **(**number**)**

**{**

**case** 1**:**

Console**.**WriteLine**(**"case 1"**);**

**goto** **case** 5**;** // переход к case 5

**case** 3**:**

Console**.**WriteLine**(**"case 3"**);**

**break;**

**case** 5**:**

Console**.**WriteLine**(**"case 5"**);**

**break;**

**default:**

Console**.**WriteLine**(**"default"**);**

**break;**

**}**

int DoOperation**(**int op**,** int a**,** int b**)**

**{**

**switch** **(**op**)**

**{**

**case** 1**:** **return** a **+** b**;**

**case** 2**:** **return** a **-** b**;**

**case** 3**:** **return** a **\*** b**;**

**default:** **return** 0**;**

**}**

**}**

int result **=** op **switch** **{**

1 **=>** a **+** b**,**

2 **=>** a **-** b**,**

3 **=>** a **\*** b**,**

\_ **=>** 0

**};**

# Перечисления enum

enum DayTime

**{**

Morning**,**

Afternoon**,**

Evening**,**

Night

**}**

DayTime**.**Morning **-** обращение к значению Morning

enum Time **:** byte

**{**

Morning**,** // 0

Afternoon**,** // 1

Evening**,** // 2

Night // 3

**}**

**(**int**)** DayTime**.**Night **-** 3

enum DayTime

**{**

Morning **=** 3**,** // каждый следующий элемент по умолчанию увеличивается на единицу

Afternoon**,** // 4

Evening**,** // 5

Night **=** Morning // 3

**}**