ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР
ПРОГРАММИРОВАНИЯ И ВЫСОКИХ ТЕХНОЛОГИЙ

2020г.

Операторы языка C#

AGENDA

- ✓ Арифметические
- ✓ Отношения
- ✓ Логические
- ✓ Присваивания
- ✓ Поразрядные
- ✓ Оператор
- ✓ Использование скобок
- ✓ Деление

Оператор

Оператор – это символ, который сообщает компилятору выполнить определенные математические или логические манипуляции.

C # имеет богатый набор встроенных операторов и предоставляет следующие типы операторов:

- арифметические;
- отношения;
- логические;
- присваивания;
- поразрядные.

Операции бывают унарными (выполняются над одним операндом), бинарными – над двумя операндами и тернарными – выполняются над тремя операндами.

Оператор

Приоритет оператора определяет группировку операций в выражении. Это влияет на оценку выражения. Некоторые операторы имеют более высокий приоритет, чем другие, например, оператор умножения имеет более высокий приоритет, чем оператор сложения.

Когда операции имеют один и тот же приоритет, порядок вычисления определяется ассоциативностью операторов. В зависимости от ассоциативности есть два типа операторов:

- Левоассоциативные операторы, которые выполняются слева направо.
- Правоассоциативные операторы, которые выполняются справа налево.

Порядок вычисления, определяемый приоритетом и ассоциативностью операторов, можно изменить с помощью скобок «**( )**».

Основные (первичные) операторы

x.m	Доступ к элементу типа.
x(…)	Вызов методов и делегатов.
x[…]	Доступ к элементу массива или ндексатора.
new T(…)	Создание объекта или делегата типа **T.**
new T[…]	Создание массива элементов типа **T.**
typeof(T)	Получение для типа **T** объекта **System.Type.**
checked(x)	Вычисление в контролируемом контексте.
unchecked(x)	Вычисление в неконтролируемом контексте.

Арифметические операторы

+	Сложение;
-	Вычитание, унарный минус;
*****	Умножение;
/	Деление;
%	Деление по модулю;
--	Декремент;
++	Инкремент.

Две формы инкремента и декремента:

- Префиксная форма (**++x, --y**).
- Постфиксная форма (**x++, y--**).

Арифметические операторы

При делении стоит учитывать, что если оба операнда представляют целые числа, то результат также будет округляться до целого числа:

```
double z = 10/4;      //результат равен 2
```

Несмотря на то, что результат операции в итоге помещается в переменную типа **double**, которая позволяет сохранить дробную часть, но в самой опеации участвуют два литерала, которые по умолчанию рассматриваются как объекты **int**, то есть целые числа, и результат то же будет целочисленный.

Для выхода из этой ситуации необходимо определять литералы или переменные, участвующие в операции, именно как типы **double** или **float**:

```
double z = 10.0/4.0;    //результат равен 2.5
```

Операторы отношения

==	Равно;
!=	Не равно;
>	Больше;
<	Меньше;
>=	Больше или равно;
<=	Меньше или равно.

Результат выполнения: **bool (true|false).**
==, != – используются для всех объектов (сравнение).
>,<,>=,<= – используеются только для объектов подерживающих отношение порядка.

Логические операторы

&	Логическое И;
|	Логическое ИЛИ;
^	Логическое исключающее ИЛИ;
&&	Логическое укороченное И;
||	Логическое укороченное ИЛИ;
!	Логическое НЕ;

Результат выполнения: **bool (true|false).**
Операнды: **bool (true|false).**

Операторы присваивания

Оператор присваивания обозначается одиночным знаком равенства (=).

Общая форма:
имя_переменной = выражение;

<имя_переменной> должно быть совместимо с типом выражения.

Составные операторы:

Оператор	**Аналог**
+=	x = x + 1;
-=	x = x - 1;
***=**	x = x*1;
/=	x = x/1;
%=	x = x%1;
|=	x = x | 1;
^=	x = x^1.

Поразрядные операторы

&	Поразрядное И;
|	Поразрядное ИЛИ;
^	Порязрядное исключающее ИЛИ;
<<	Сдвиг влево;
>>	Сдвиг вправо;
~	Дополнение до 1 (унарный оператор НЕ);
<<	Сдвиг влево (в сторону старших бит);
>>	Сдвиг вправо бит (в сторону младших бит).

Работают с отдельными битами значения.
Используются только для целочисленных переменных.

Поразрядные операторы

p	q	p & q	p | q	p ^ q	~p
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

«**and**» – проверка наличия установленных в единицу битов или осуществление обнуления некоторых битов;

«**or**» – установка в единицу отдельных битов;

«**xor**» – смена значения бита (или нескольких битов) на противоположное.