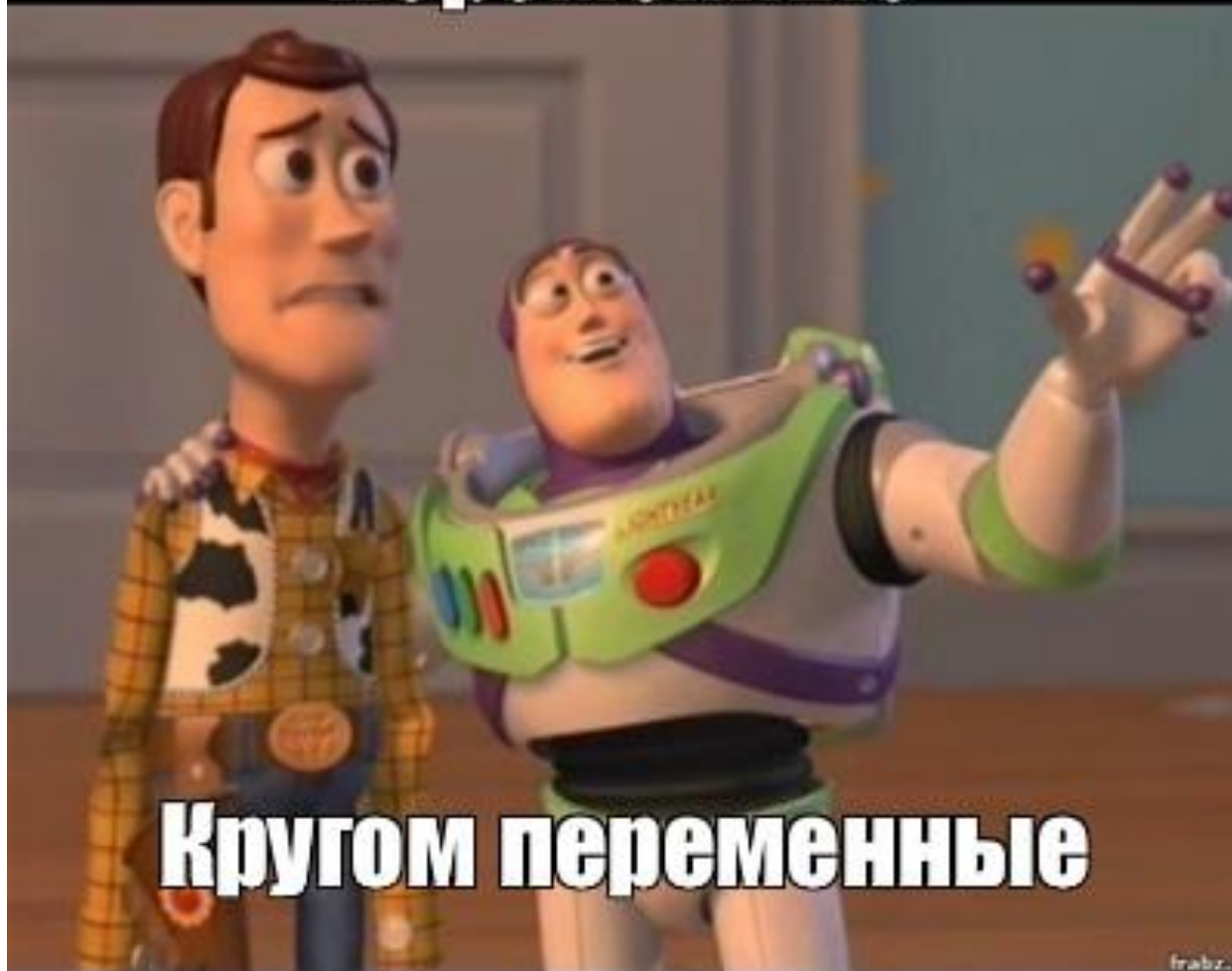


# Переменные

Арифметические операции

**Переменные**



**Кругом переменные**



Логический тип			
Имя типа в C#	Системный тип	Значения	Размер
<b>bool</b>	<b>Boolean</b>	true, false	8 бит
Арифметические целочисленные типы			
Имя типа	Системный тип	Диапазон	Размер
<b>sbyte</b>	<b>SByte</b>	-128 — 128	Знаковое, 8 Бит
<b>byte</b>	<b>Byte</b>	0 — 255	Беззнаковое, 8 Бит
<b>short</b>	<b>Short</b>	-32768 — 32767	Знаковое, 16 Бит
<b>ushort</b>	<b>UShort</b>	0 — 65535	Беззнаковое, 16 Бит
<b>int</b>	<b>Int32</b>	$\approx(-2 \cdot 10^9 \text{ — } 2 \cdot 10^9)$	Знаковое, 32 Бит
<b>uint</b>	<b>UInt32</b>	$\approx(0 \text{ — } 4 \cdot 10^9)$	Беззнаковое, 32 Бит
<b>long</b>	<b>Int64</b>	$\approx(-9 \cdot 10^{18} \text{ — } 9 \cdot 10^{18})$	Знаковое, 64 Бит
<b>ulong</b>	<b>UInt64</b>	$\approx(0 \text{ — } 18 \cdot 10^{18})$	Беззнаковое, 64 Бит
Арифметический тип с плавающей точкой			
Имя типа	Системный тип	Диапазон	Точность
<b>Float</b>	<b>Single</b>	$+1.5 \cdot 10^{-45} \text{ - } +3.4 \cdot 10^{38}$	7 цифр
<b>double</b>	<b>Double</b>	$+5.0 \cdot 10^{-324} \text{ - } +1.7 \cdot 10^{308}$	15-16 цифр
Арифметический тип с фиксированной точкой			
<b>decimal</b>	<b>Decimal</b>	$+1.0 \cdot 10^{-28} \text{ - } +7.9 \cdot 10^{28}$	28-29 значащих цифр
Символьные типы			
<b>char</b>	<b>Char</b>	U+0000 - U+ffff	16 бит Unicode символ
<b>string</b>	<b>String</b>	Строка из символов Unicode	
Объектный тип			
Имя типа	Системный тип	Примечание	
<b>object</b>	<b>Object</b>	Прародитель всех встроенных и пользовательских типов	

C# предоставляет ряд операторов. Многие из них поддерживаются встроенными типами и позволяют выполнять базовые операции со значениями этих типов.

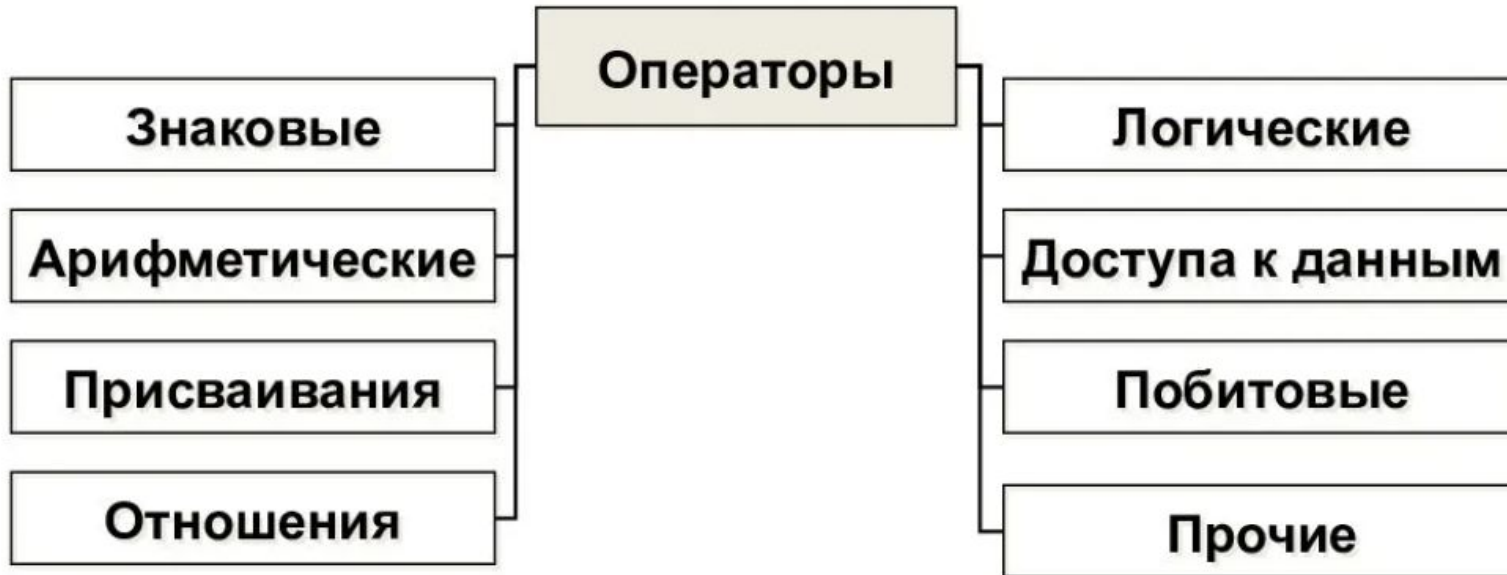
- Арифметические операторы, выполняющие арифметические операции с числовыми операндами.
- Операторы сравнения, сравнивающие числовые операнды.
- Логические операторы, выполняющие логические операции с операндами `bool`.
- Битовые операторы и операторы сдвига выполняют битовые операции или операции сдвига с операндами целочисленных типов.
- Операторы равенства проверяют равенство или неравенство своих операндов.

# Арифметические операции



**Такая вот арифметика**

**Оператор** – это команда языка программирования высокого уровня.



Операторы можно разделить на группы в зависимости от того, с каким количеством операндов они работают:  
**унарные, бинарные и тернарные**

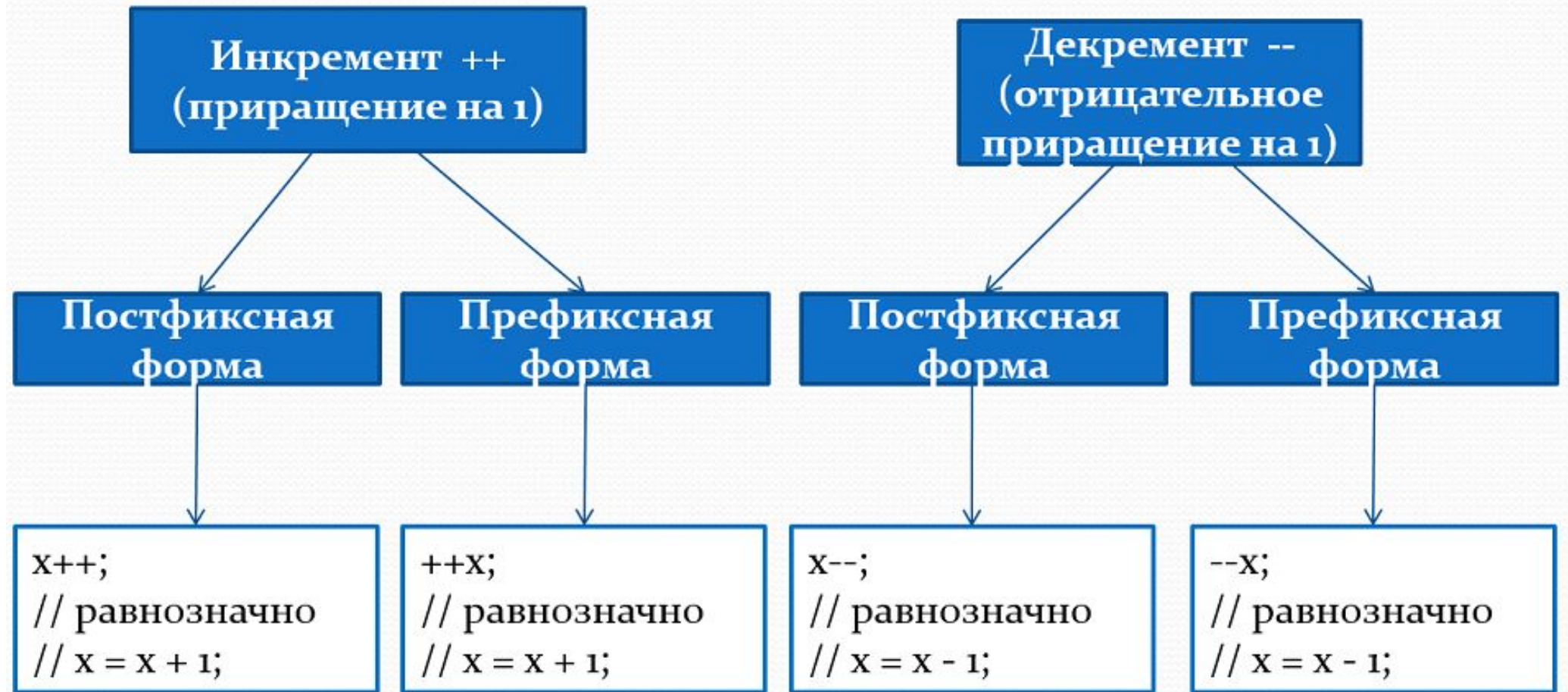


# Следующие операторы выполняют арифметические операции с операндами числовых типов:

- унарные — ++ (приращение), -- (уменьшение), + (плюс) и - (минус);
- бинарные — \* (умножение), / (деление), % (остаток от деления), + (сложение) и - (вычитание).

Эти операторы поддерживаются всеми целочисленными типами и типами с плавающей запятой.

# Инкремент(++ ) и декремент(-- )



# В чем разница префиксного и постфиксного инкремента, декремента?

2.

Оператор	Символ	Пример	Операция
Префиксный инкремент (пре-инкремент)	++	++x	Инкремент x, затем вычисление x
Префиксный декремент (пре-декремент)	--	--x	Декремент x, затем вычисление x
Постфиксный инкремент (пост-инкремент)	++	x++	Вычисление x, затем инкремент x
Постфиксный декремент (пост-декремент)	--	x--	Вычисление x, затем декремент x

# Пример 1

```
int i = 0;
i++; // i равно 1
int t1 = i++; // t1 равно 1, i равно 2
int t2 = --i; // t2 равно 1, i равно 1
for (int k = 1; k <= 5; k++)
{
    Console.WriteLine(k);
} // Цикл выведет целые числа от 1 до 5
```

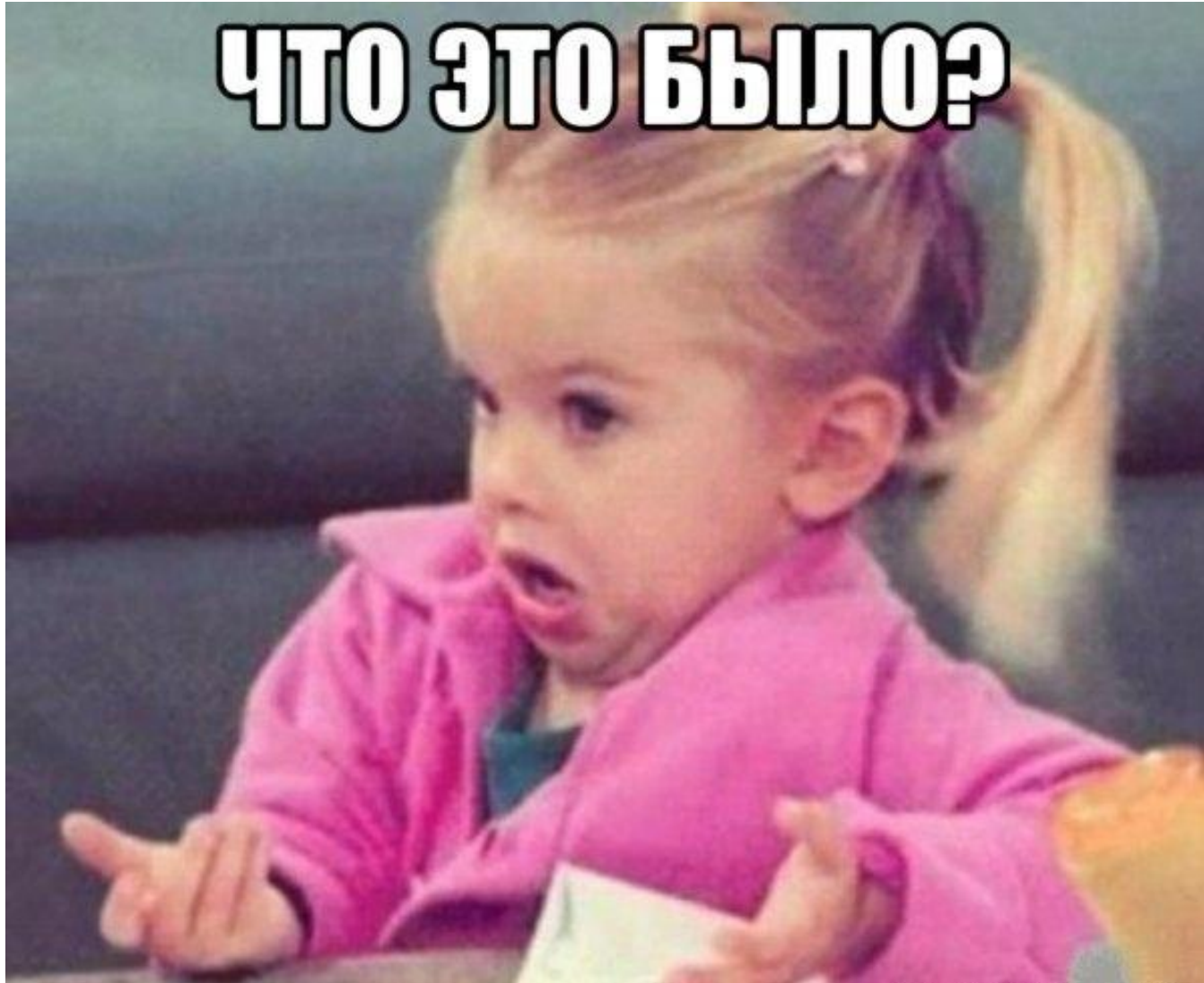
Операторы инкрементации и декрементации следует использовать везде, где необходимо единообразно изменить значение некой числовой переменной на единицу. Они задействуют механизмы, отличные от обычного суммирования или вычитания, таким образом являясь более эффективными.

Выражения с этими операторами можно вставлять в код независимо, тогда неважно, с какой стороны от операнда стоит оператор. Если же такая операция происходит внутри выражения, то появляется принципиальная разница между префиксным и постфиксным способом.

## Пример 2. Чему равно a и b?

```
int exemp = 0;  
    int a = exemp++;  
    int b = ++exemp;  
    Debug.Log("a="+a);  
    Debug.Log("b=" + b);
```

**ЧТО ЭТО БЫЛО?**



# Операторы унарного плюса и минуса

## Знаковые операторы (унарный «+» и унарный «-»)

4

Оператор	Назначение
+	Значение операнда без изменения знака
-	Значение операнда с противоположным знаком

Унарными операторами "+" и "-" можно присваивать знаки величинам арифметических типов. Если перед величиной не указан знак, то значение по умолчанию считается положительным.

### Пример кода

```
int a = -5;  
float b = +45.67; // float b = 45.67;  
double c = -b;    // c <- -45.67
```



# Арифметические операторы

Бинарные операторы	Назначение
+	Суммирует два операнда
–	Вычитает из первого операнда второй
*	Умножает два операнда
/	Делит первый операнд на второй
%	Остаток от целочисленного деления первого операнда на второй

Операции умножения, деления и остатка имеют более высокий приоритет, чем операции сложения и вычитания. При равном приоритете операторы обрабатываются в последовательности слева направо. Можно изменить порядок выполнения операторов с помощью скобок.

<i>Знак операции</i>	<i>Назначение</i>	<i>Пример использования</i>	<i>Результат</i>
+	сложение	2+5	7
-	вычитание	4-1	3
*	умножение	3*5	15
/	деление (обычное)	2.4/2	1.2
/	целочисленное деление	5/2	2
%	<p>вычисление остатка при целочисленном делении</p> <p><u>Внимание!</u></p> <p>Операция вычисления остатка (%) применима только для <u>целочисленных</u> операндов.</p>	5%2	1