В C# используется большинство операций, которые применяются и в других языках программирования. Операции представляют определенные действия над операндами - участниками операции. В качестве операнда может выступать переменной или какое-либо значение (например, число). Операции бывают унарными (выполняются над одним операндом), бинарными - над двумя операндами и тернарными - выполняются над тремя операндами. Рассмотрим все виды операций.

Бинарные арифметические операции:

- **+**

  Операция сложения двух чисел:

```
int x = 10;
int z = x + 12; // 22
```

- **-**

  Операция вычитания двух чисел:

```
int x = 10;
int z = x - 6; // 4
```

- *****

  Операция умножения двух чисел:

```
int x = 10;
int z = x * 5; // 50
```

- ***/***

  операция деления двух чисел:

```
int x = 10;
int z = x / 5; // 2

double a = 10;
double b = 3;
double c = a / b; // 3.33333333
```

  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 операции 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
```

- **%**

  Операция получение остатка от целочисленного деления двух чисел:

```
double x = 10.0;
double z = x % 4.0; //результат равен 2
```

Также есть ряд унарных операций, в которых принимает участие один операнд:

- **++**

  Операция инкремента

  Инкремент бывает префиксным: `++x` - сначала значение переменной x увеличивается на 1, а потом ее значение возвращается в качестве результата операции.

  И также существует постфиксный инкремент: `x++` - сначала значение переменной x возвращается в качестве результата операции, а затем к нему прибавляется 1.

```
int x1 = 5;
int z1 = ++x1; // z1=6; x1=6
Console.WriteLine($"{x1} - {z1}");

int x2 = 5;
int z2 = x2++; // z2=5; x2=6
Console.WriteLine($"{x2} - {z2}");
```

- **--**

  Операция декремента или уменьшения значения на единицу. Также существует префиксная форма декремента (`--x`) и постфиксная (`x--`).

```
int x1 = 5;
int z1 = --x1; // z1=4; x1=4
Console.WriteLine($"{x1} - {z1}");

int x2 = 5;
int z2 = x2--; // z2=5; x2=4
Console.WriteLine($"{x2} - {z2}");
```

При выполнении сразу нескольких арифметических операций следует учитывать порядок их выполнения. Приоритет операций от наивысшего к низшему:

1. Инкремент, декремент
2. Умножение, деление, получение остатка
3. Сложение, вычитание

Для изменения порядка следования операций применяются скобки.

Рассмотрим набор операций:

```
int a = 3;
int b = 5;
int c = 40;
int d = c---b*a;    // a=3  b=5  c=39  d=25
Console.WriteLine($"a={a}  b={b}  c={c}  d={d}");
```

Здесь мы имеем дело с тремя операциями: декремент, вычитание и умножение. Сначала выполняется декремент переменной c, затем умножение b*a, и в конце вычитание. То есть фактически набор операций выглядел так:

```
int d = (c--)-(b*a);
```

Но с помощью скобок мы могли бы изменить порядок операций, например, следующим образом:

```
int a = 3;
int b = 5;
int c = 40;
int d = (c-(--b))*a;    // a=3  b=4  c=40  d=108
Console.WriteLine($"a={a}  b={b}  c={c}  d={d}");
```

Ассоциативность операторов

Как выше было отмечено, операции умножения и деления имеют один и тот же приоритет, но какой тогда результат будет в выражении:

```
int x = 10 / 5 * 2;
```

Стоит нам трактовать это выражение как `(10 / 5) * 2` или как `10 / (5 * 2)`? Ведь в зависимости от трактовки мы получим разные результаты.

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

Все арифметические операторы являются левоассоциативными, то есть выполняются слева направо. Поэтому выражение `10 / 5 * 2` необходимо трактовать как `(10 / 5) * 2`, то есть результатом будет 4.