ОПЕРАТОРНЫЕ МЕТОДЫ

подробно обо всем

Можем с помощью оператора **+** сложить два числа и получить в результате число. Мы можем сложить с помощью того же оператора два текстовых значения и получить в результате текст.

Имеется ограниченное количество ситуаций, в которых могут использоваться операторы языка C#, а результат выполнения операций зависит от типа операндов

В языке C# очень мощный и эффективный механизм, который связан с ***перегрузкой операторов***.

Основная идея в том, что для **объектов пользовательских классов** (то есть классов, которые мы описываем в программе) **доопределяется действие встроенных** операторов языка C#

Описывая класс, возможно путем ***несложных манипуляций*** добиться того, что объекты этого класса можно будет складывать, как числа.

Чтобы определить способ применения того или иного оператора к объектам класса, в классе описывается специальный метод, который называется ***операторным***

Название операторного метода состоит из ключевого слова ***operator*** и символа оператора, для которого определяется способ применения к объектам класса

Например, если мы хотим определить операцию сложения для объектов класса с помощью оператора +, то в классе нужно описать операторный метод с названием ***operator+.***

Если в классе описан операторный метод с названием ***operator****

1) Операторный метод описывается в классе, для объектов которого
определяется действие соответствующего оператора.

2) Операторный метод описывается как статический (то есть с ключевым словом static) и открытый (с ключевым словом public).

3) Операторный метод должен возвращать результат

4) Аргументы операторного метода отождествляются с операндами выражения, обрабатываемого операторным методом.

В языке C# перегружаться могут многие операторы, но не все. Например, среди бинарных операторов для перегрузки доступны такие:
+, -, *, /, %, &, [, ^, < < и >>

Перегружать можно и операторы сравнения, но такие операторы должны перегружаться парами: == и ! =, < и >, <= и >=

Не перегружаются сокращенные операторы присваивания, такие как +=, -=, *=, /=, %=, & =, ^=, <<= и >>=.

Есть операторы, которые не перегружаются. Среди них:
= (присваивание),
. (оператор точка»),
?: (тернарный оператор),
-> (оператор стрелка), используемый при работе с указателями на структуры),
=> (оператор стрелка), используемый при описании лямбда-выражений),
new (оператор создания нового объекта)

Категория	Операция
Арифметические	`+ - * / %`
Логические	`& | ^ ~ && || !`
Конкатенация строк	`+`
Инкремент и декремент	`++ --`
Побитовый сдвиг	`<< >>`
Сравнение	`== != < > <= >=`
Присваивание	`= += -= *= /= %= &= |= ^= <<= >>=`
Доступ к членам (для объектов)	`.`
Индексирование (для массивов и индексаторов)	`[]`
Преобразование типа	`()`
Условная (тернарная операция)	`?:`
Создание объекта	`new`
Информация о типе	`sizeof` (только небезопасный код) `is typeof`
Управление исключениями переполнения	`checked unchecked`
Разыменование и адресация	`* -> &` (только небезопасный код) `[]`

- неявное преобразование (автоматическое) - преобразованием меньше по размеру типа данных в больший: char -> int -> long -> float -> double
- явное преобразование (ручное) - обратное предыдущему преобразование большего типа в меньший: double -> float -> long -> int -> char

```
//Класс с перегрузкой оператора сложения :
Ссылок: 7
class MyClass
{
    //Целочисленное поле :
    public int number;
    //Конструктор с целочисленным аргументом :
    Ссылок: 2
    public MyClass(int n)
    {
        //Присваивание значения полю:
        number = n;
    }
    //Операторный метод для перегрузки оператора сложения :
    Ссылок: 1
    public static int operator +(MyClass a, MyClass b)
    {
        //Локальная целочисленная переменная :
        int m = a.number + b.number;
        // Результат метода :
        return m;
    }
}
    //Класс с главным методом :
    Ссылок: 0
    class OverloadingOperatorPlusDemo
    {
    //Главный метод :
        Ссылок: 0
        static void Main()
        {
            //Создание объектов класса MyClass :
            MyClass A = new MyClass(100);
            MyClass B = new MyClass(200);
            //Целочисленная переменная :
            int num;
            num = A + B;
            Console.WriteLine("A+B=" + num);
        }
    }
}
```