При описі класів, що характеризують поведінку математичних об’єктів, зручно було б використовувати традиційні математичні знаки операцій для виконання відповідних дій. Наприклад, при додаванні двох комплексних чисел природніше було б використовувати оператор «+», а не викликати, припустимо, метод Sum(). Для таких ситуацій у мові C# є зручний засіб, що називається **перевантаженням операторів**.

Унарні оператори. У мові C# у класах можна перевантажувати наступні унарні оператори:

+ - ! ~ ++ -- true false

Загальне правило перевантаження унарного оператора:

```
public static <тип результату> operator <унарний оператор>
     <ім'я класу> <аргумент>)
{<тіло оператора>}
```

Аргумент, який передається в оператор повинен мати тип класу, для якого цей оператор перевантажується.

Унарний оператор повинен повертати:

- для операторів +,-,!,~ величину будь-якого типу;
- для операторів ++, -- величину типу класу, для якого цей оператор перевантажується;
- для операторів true, false величину типу bool.

Унарний оператор не повинен змінювати значення аргументу. Унарний оператор, який повертає величину типу класу, для якого цей оператор перевантажується, повинен створювати новий об’єкт цього класу, виконати з ним необхідні дії і повернути його в якості результату.

Унарні оператори `true`, `false` повинні перевантажуватись разом.

```
public static Parallelepiped operator ++(Parallelepiped p)
        {return new Parallelepiped(p.A + 1, p.B + 1, p.H + 1);
        }
```

Процедура обробки події натиснення кнопки *Інкрементації паралелепіпеда*:

```
private void button8_Click(object sender, EventArgs e)
 {var p = new Parallelepiped(5, 4, 8); p.Info();
  p++; p.Info();
  ++p; p.Info();
 }
```

Бінарні оператори. У мові C# у класах можна перевантажувати наступні бінарні оператори:

+ - * / % & | ^ << >> == != > < >= <=

Загальне правило перевантаження бінарного оператора:

```
public static <тип результату> operator <бінарна операція>
     <тип 1> <арг. 1>, <тип 2> <арг. 2>)
{ <тіло оператора> }
```

Принаймні один аргумент у бінарному операторі повинен мати тип класу, для якого цей оператор перевантажується. Бінарний оператор може повертати величину будь-якого типу. Операції == та !=, > та = та <= повинні перевантажуватись парами та повертати логічне значення. Бінарний оператор, який повертає величину типу класу, для якого цей оператор перевантажується, повинен створювати новий об’єкт цього класу, виконати з ним необхідні дії і повернути його в якості результату. Складні операції присвоювання (+=, -= і т.д.) перевантажувати не можна. Але якщо у класі перевантажена відповідна бінарна операція, то при виконанні складної операції присвоювання викликаються спочатку відповідна бінарна операція, а потім операція присвоювання.

```
class Parallelepiped : Parallelogram
{…
    public double minSide()
    {return Math.Min(Math.Min(A, B), H);
    }

    public double maxSide()
    {double max = A;
```

```
        if (B > max) max = B;
        if (H > max) max = H;
        return max;
       }

       new public double perimeter()
       {return 4*(A+B+H);
       }

       public static bool operator ==(Parallelepiped p1, Parallelepiped p2)
       {return p1.minSide() == p2.minSide() &&
               p1.maxSide() == p2.maxSide() &&
               p1.perimeter() == p2.perimeter();
       }

       public static bool operator !=(Parallelepiped p1, Parallelepiped p2)
       {return !(p1 == p2);
       }
```

Процедура обробки події натиснення кнопки *Порівняння паралелепіпедів*:

```
private void button7_Click(object sender, EventArgs e)
 {   var p1 = new Parallelepiped(4, 5, 8);
     p1.Info();
     var p2 = new Parallelepiped(5, 4, 10);
     p2.Info();
     if (p1 == p2) MessageBox.Show("Паралелепіпеди рівні");
     else MessageBox.Show("Паралелепіпеди не рівні");
 }
```

Операції перетворення типів. У мові C# є можливість перевантажувати операції явного та неявного перетворення типів. Загальне правило перевантаження операції *неявного* перетворення типів:

```
public static implicit operator <цільовий тип> (<тип>
<аргумент>)
{ <тіло оператора> }
```

Загальне правило перевантаження операції *явного* перетворення типів:

```
public static explicit operator <цільовий тип>(<тип>
<аргумент>)
{ <тіло оператора> }
```

Обидві операції виконують перетворення із типу аргумента в цільовий тип. Одним із цих типів має бути тип класу, у якому перевантажується операція.

```
class Complex
{
 public static implicit operator Complex(double d)
 {return new Complex(d,0); }
 public static explicit operator double(Complex c)
 {return c.a; }
}
// неявне перетворення дійсного числа у комплексне
double d1 = 1.5;
Complex c1 = d1; // 1.5+0i
// явне перетворення комплексного числа у дійсне
Complex c2 = new Complex(-3,0);
double d2 = (double)c2; // d2=-3;
```