# **Operator overloading**

Тип, визначений користувачем, може перевантажувати попередньо визначений оператор C#. Тобто, тип може забезпечити власну реалізацію операції у випадку, якщо один або обидва операнди мають цей тип.

Для оголошення оператора використовується ключове слово operator. Оголошення оператора має відповідати наступним правилам:

* Містити модифікатор ***public*** та ***static***.
* Унарний оператор має один вхідний параметр. Бінарний оператор має два вхідних параметри.
* У кожному випадку хоча б один параметр повинен мати тип ***T*** або ***T?*** де ***T*** - тип, який містить оголошення оператора.

У наступному прикладі описано спрощену структуру для представлення раціонального числа. Структура перевантажує деякі арифметичні оператори:

public readonly struct Fraction

{

private readonly int num;

private readonly int den;

public Fraction(int numerator, int denominator)

{

if (denominator == 0)

{

throw new ArgumentException("Denominator cannot be zero.", nameof(denominator));

}

num = numerator;

den = denominator;

}

public static Fraction operator +(Fraction a) => a;

public static Fraction operator -(Fraction a) => new Fraction(-a.num, a.den);

public static Fraction operator +(Fraction a, Fraction b)

=> new Fraction(a.num \* b.den + b.num \* a.den, a.den \* b.den);

public static Fraction operator -(Fraction a, Fraction b)

=> a + (-b);

public static Fraction operator \*(Fraction a, Fraction b)

=> new Fraction(a.num \* b.num, a.den \* b.den);

public static Fraction operator /(Fraction a, Fraction b)

{

if (b.num == 0)

{

throw new DivideByZeroException();

}

return new Fraction(a.num \* b.den, a.den \* b.num);

}

public override string ToString() => $"{num} / {den}";

}

public static class OperatorOverloading

{

public static void Main()

{

var a = new Fraction(5, 4);

var b = new Fraction(1, 2);

Console.WriteLine(-a); // output: -5 / 4

Console.WriteLine(a + b); // output: 14 / 8

Console.WriteLine(a - b); // output: 6 / 8

Console.WriteLine(a \* b); // output: 5 / 8

Console.WriteLine(a / b); // output: 10 / 4

}

}

Ви можете розширити попередній приклад, визначивши неявне перетворення з ***int*** у ***Fraction***. Тоді перевантажені оператори підтримували б аргументи цих двох типів. Тобто, стало б можливим додати ціле число до дробу і отримати в результаті дріб.

Ви також можете використовувати ключове слово operator для визначення користувацького перетворення типів. Для отримання додаткової інформації див. статтю Оператори перетворення, визначені користувачем.

## **Overloadable operators**

The following table shows the operators that can be overloaded:

| **Operators** | **Notes** |
| --- | --- |
| **+x, -x, !x, ~x, ++, --, true, false** | The true and false operators must be overloaded together.  Оператори ***true*** та ***false*** повинні бути перевантажені разом. |
| **x + y, x - y, x \* y, x / y, x % y,**  **x & y, x | y, x ^ y,**  **x << y, x >> y, x >>> y** |  |
| **x == y, x != y, x < y, x > y, x <= y, x >= y** | Must be overloaded in pairs as follows: == and !=, < and >, <= and >=. |

## **Non overloadable operators**

The following table shows the operators that can't be overloaded:

| **Operators** | **Alternatives** |
| --- | --- |
| x && y, x || y | Overload both the **true** and **false** operators and the ***&*** or ***|*** operators. For more information, see User-defined conditional logical operators.  *Перевантажуйте оператори* ***true*** *і* ***false****, а також оператори* ***&*** *і* ***|****. Докладні відомості наведено у статті Умовні логічні оператори, визначені користувачем.* |
| [a[i]](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#indexer-operator-), [a?[i]](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#null-conditional-operators--and-) | Define an indexer.  *Визначте індексатор.* |
| [(T)x](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/type-testing-and-cast#cast-expression) | Define custom type conversions that can be performed by a cast expression.  *Визначте кастомні перетворення типів, які можуть бути виконані за допомогою приведення.* |
| [+=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [-=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [\*=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [/=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [%=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [&=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators#compound-assignment), [|=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators#compound-assignment), [^=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators#compound-assignment), [<<=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#compound-assignment), [>>=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#compound-assignment), [>>>=](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#compound-assignment) | Overload the corresponding binary operator. For example, when you overload the binary + operator, += is implicitly overloaded.  *Перевантажити відповідний бінарний оператор. Наприклад, коли ви перевантажуєте бінарний оператор* ***+****, неявно перевантажується* ***+=****.* |
| [^x](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#index-from-end-operator-), [x = y](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/assignment-operator), [x.y](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#member-access-expression-), [x?.y](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#null-conditional-operators--and-), [c ? t : f](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/conditional-operator), [x ?? y](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-coalescing-operator), [??= y](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-coalescing-operator),  [x..y](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#range-operator-), [x->y](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/pointer-related-operators#pointer-member-access-operator--), [=>](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-operator), [f(x)](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#invocation-expression-), [as](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/type-testing-and-cast#as-operator), [await](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/await), [checked](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/checked-and-unchecked), [unchecked](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/checked-and-unchecked), [default](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/default), [delegate](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/delegate-operator), [is](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/type-testing-and-cast#is-operator), [nameof](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/nameof), [new](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/new-operator),  [sizeof](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/sizeof), [stackalloc](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/stackalloc), [switch](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/switch-expression), [typeof](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/type-testing-and-cast#typeof-operator), [with](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/with-expression) | None |