# NẠP CHỒNG TOÁN TỬ

**Mục tiêu:** Sau khi tìm hiểu xong chương này người học có thể nắm được các nội dung sau:

* Đưa ra được các nguyên tắc nạp chồng toán tử.
* Biết được các yêu cầu cần thiết khi sử dụng toán tử.
* Nạp chồng được các hàm toán tử tiêu biểu (+, -, =, !=, ==, [],…)
* Chuyển đổi kiểu ngầm định trong kiểu dữ liệu đối tượng.

# A. TÓM TẮT LÝ THUYẾT

1. Toán tử

Toán tử được kí hiệu bằng một biểu tượng dùng để thực hiện một hành động. Các kiểu dữ liệu cơ bản của C# như kiểu số nguyên hỗ trợ rất nhiều các toán tử gán, toán tử toán học, toán tử logic,…

1. Nạp chồng toán tử

Toán tử được nạp chồng bằng cách định nghĩa một phương thức toán tử, phương thức toán tử là phương thức tĩnh, giá trị trả về của nó thể hiện kết quả của một phép toán và những tham số là toán hạng. Khi chúng ta tạo một toán tử cho một lớp là chúng ta nạp chồng toán tử (operator overloading), cũng giống như là chúng ta nạp chồng bất cứ phương thức thành viên nào.

Việc nạp chồng toán tử thực chất cũng giống như ta xây dựng một phương thức thông thường và kết quả thu được như nhau nhưng nạp chồng toán tử giúp người lập trình có thể sử dụng các kí hiệu toán học thường gặp (+, -, \*, /), dễ nhớ và gần gũi hơn.

1. Cú pháp nạp chồng toán tử

Phương thức toán tử bao gồm từ khóa operator theo sau là kí hiệu của toán tử được nạp chồng.

C# cho phép nạp chồng toán tử như sau:

*Type operator operator\_symbol(parameter\_list);*

Trong đó:

- Type là kiểu giá trị trả về của phương thức

- parameter\_list là danh sách các đối số nếu có

- Operator\_symbol là các toán tử.

Trong C# có các hàm toán tử có thể nạp chồng và các phương thức thay thế như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Toán tử | Tên phương thức thay thế | Toán tử | Tên phương thức thay thế |
| + | Add | > | Compare |
| - | Subtract | < | Compare |
| \* | Multiply | != | Compare |
| / | Divide | >= | Compare |
| % | Mod | <= | Compare |
| ^ | Xor | \*= | Multiply |
| & | BitwiseAnd | - = | Subtract |
| | | Bitwiseor | ^= | Xor |
| && | Add | <<= | leftshift |
| || | Or | %= | Mod |
| = | Assign | += | Add |
| << | leftshift | &= | BitwiseAnd |
| >> | Rightshift | |= | Bitwiseor |
| = = | Equals | /= | Divide |
| -- | Decrement | - | Negate |
| ++ | Increment |  |  |

**Ví dụ 1:** Sử dụng nạp chồng toán tử + để cộng hai phân số

public static phanso operator +(phanso rhs, phanso lhs)

{

//các câu lệnh

}

Giải thích: rhs là phân số bên phải, lhs là phân số bên trái.

**Ví dụ 2:** Sử dụng nạp chồng toán tử - một ngôi để đổi dấu cho một số như sau:

public static So operator –(So s)

{

}

1. Toán tử đơn phân

Các toán tử dùng cho một đối tượng hay tham chiếu đến đối tượng trong lớp. Ví dụ các toán tử đơn phân như + ( +a ), - (-a), ++ (++a),…

1. Toán tử nhị phân

Các toán tử như toán tử == so sánh giữa hai đối tượng. toán tử != so sánh không bằng giữa hai đối tượng, < so sánh nhỏ hơn, > so sánh lớn hơn,… Đây là các toán tử phải có các cặp toán hạng hay ta còn gọi là toán tử nhị phân.

1. Hỗ trợ ngôn ngữ .NET khác

C# cung cấp khả năng cho phép nạp chồng toán tử cho các lớp mà chúng ta xây dựng. Trong khi đó các ngôn ngữ .NET khác không cho phép điều này.

+ Để đảm bảo lớp hỗ trợ các phương thức thay thế cho phép các ngôn ngữ khác có thể gọi. Để làm được điều này thì khi nạp chồng toán tử nào đó thì phải xây dựng cho nó một phương thức có chức năng tương tự như toán tử đã chồng.

**Ví dụ:** Khi chúng ta nạp chồng toán tử + thì phải cung cấp một phương thức Add( ) cũng làm chức năng cộng hai đối tượng.

Khi sử dụng chồng toán tử == thì nên cung cấp thêm phương thức Equals() bởi đối tượng và hướng chức năng này đến toán tử == cho phép lớp của ta đa hình và cung cấp khả năng hữu ích cho ngôn ngữ .Net khác.

+ Toán tử so sánh: Có 6 toán tử so sánh ứng với 3 cặp. Giữa các cặp này luôn luôn có kết quả đối nghịch nhau: Nếu toán hạng đầu trả về giá trị true thì toán hạng kia trả về giá trị false. C# luôn luôn yêu cầu nạp chồng cả hai toán tử đó. Nếu nạp chồng toán tử == thì phải nạp chồng toán tử !=, nếu không trình biên dịch sẽ báo lỗi.

Có một hạn chế là toán tử so sánh phải trả về kiểu bool, và đó cũng là điểm khác nhau giữa các toán hạng này và toán hạng số học.

Cấu trúc của phương thức có chức năng tương tự như toán tử so sánh đã nạp chồng được xây dựng như sau:

public override bool Equals(object obj)

{

Console.WriteLine("phuong thuc equals");

if (!(obj is phanso))

{

return false;

}

return this = = (phanso)obj;

}

Trong đó toán tử is dùng để kiểm tra kiểu đối tượng lúc chương trình đang thực hiện có thích ứng với toán hạng hay không. Nếu kiểm tra thấy kiểu đối tượng thích ứng với toán hạng thì kết quả trả ra là true và ngược lại là false.

+ Toán tử chuyển đổi: Dùng từ khóa implicit và explicit nó có tác dụng chuyển đổi một số từ kiểu có kích thước nhỏ sang kích thước lớn và (ngược lại) mà không bị mất thông tin.

1. Một số trường hợp nên sử dụng các toán tử nạp chồng

* Trong toán học, mọi đối tượng toán học như tọa độ, vector, ma trận, hàm số,… Nếu bạn viết chương trình làm những mô hình toán học hay vật lý, bạn nhất định sẽ mô tả những đối tượng này.
* Những chương trình đồ họa sẽ sử dụng các đối tượng toán học và tọa độ khi tính toán vị trí của trên màn hình.
* Một lớp mô tả số lượng tiền.
* Việc xử lý từ hay chương trình phân tích văn bản có lớp để mô tả các câu văn, mệnh đề và bạn phải sử dụng các toán hạng để liên kết các câu lại với nhau.

1. Yêu cầu khi sử dụng toán tử

+ Định nghĩa những toán tử trong kiểu dữ liệu giá trị, kiểu dữ liệu xây dựng sẵn.

+ Cung cấp phương thức nạp chồng toán tử chỉ bên trong của lớp nơi mà những phương thức được định nghĩa.

+ Sử dụng tên và các kí hiệu được mô tả trong trong Common Language Specification (CLS), tức là ta không được nạp chồng một toán tử mới.

+ Sử dụng chồng toán tử trong trường hợp kết quả trả về rõ ràng.

**Ví dụ:**

Khi ta sử dụng toán tử or hoặc and giữa một giá trị thời gian này với một thời gian khác thì kết quả trả về sẽ không rõ ràng, vì vậy trong trường hợp này ta không nên sử dụng nạp chồng toán tử.

+ Nạp chồng toán tử có tính chất đối xứng. Nghĩa là:

Khi dùng toán tử (==) thì phải dùng toán tử (!=)

Khi dùng toán tử (<) thì phải dùng toán tử (>).

Khi dùng toán tử (>=) thì phải dùng toán tử (<=).

+ Phải cung cấp các phương thức thay thế cho toán tử được nạp chồng. Bởi vì hầu hết các ngôn ngữ khác không cho phép chồng toán tử nên khi ta sử dụng thêm phương thức có chức năng tương tự như toán tử giúp cho chương trình có thể phù hợp với nhiều ngôn ngữ khác nhau.

1. Ưu và nhược điểm của chồng toán tử

Ưu điểm:

+ Nạp chồng toán tử là một phương thức ngắn gọn giúp mã nguồn dễ nhìn hơn, dễ quản lí, sáng sủa hơn.

+ Nạp chồng toán tử là đường dẫn cho các đối tượng.

Nhược điểm:

+ Ta phải sử dụng nạp chồng toán tử một cách hạn chế và phù hợp với các toán tử của lớp được xây dựng sẵn, không sử dụng toán tử quá mới hay quá riêng rẽ. Nếu sử dụng toán tử một cách lạm dụng thì sẽ làm chương trình nhầm lẫn.

Console.WriteLine("---------------Ham main bai 3--------------------");

PhanSo a = new PhanSo(3, 7);

PhanSo b = new PhanSo(4, 9);

Console.WriteLine("Phan so a: ");

a.InPhanSo();

Console.WriteLine("Phan so b: ");

b.InPhanSo();

PhanSo x = new PhanSo();

PhanSo y = new PhanSo();

Console.WriteLine("Nhap phan so x:");

x.NhapPhanSo();

Console.WriteLine("Nhap phan so y:");

y.NhapPhanSo();

Console.WriteLine("Gia tri nghich dao cu phan so x:");

x.GiaTriNghicDao().InPhanSo();

Console.WriteLine("Tong x + y:");

PhanSo tong = x + y;

tong.InPhanSo();

Console.Write("Nhap so luong phan so n: ");

do

{

n = Int32.Parse(Console.ReadLine());

} while (n <= 0);

PhanSo[] dsPhanSo = new PhanSo[n];

for (int i = 0; i < n; i++)

{

Console.WriteLine("Nhap phan so thu {0}:", i + 1);

dsPhanSo[i] = new PhanSo();

dsPhanSo[i].NhapPhanSo();

}

tong = new PhanSo(0, 1);

for (int i = 0; i < n; i++)

{

tong = tong + dsPhanSo[i];

}

Console.WriteLine("Tong cu {0} phan so:", n);

tong.InPhanSo();

PhanSo max = dsPhanSo[0];

int vitri = 0;

for (int i = 0; i < n; i++)

{

if (max < dsPhanSo[i])

{

max = dsPhanSo[i];

vitri = i;

}

}

Console.WriteLine("Phan so lon nhat la:");

max.InPhanSo();

Console.WriteLine("Phan so lon nhat tai vi tri: {0}\n", vitri);

PhanSo.SapXep(dsPhanSo);

Console.WriteLine("Danh sach phan so da sap xep tang dan:");

for (int i = 0; i < n; i++)

{

dsPhanSo[i].InPhanSo();

}

Console.WriteLine("Danh sach phan so da sap xep giam dan:");

PhanSo.SapXep(dsPhanSo, false);

for (int i = 0; i < n; i++)

{

dsPhanSo[i].InPhanSo();

}

Console.ReadKey();

Console.ReadKey();

}