Trường Đại học Công Thương Tp. Hồ Chí Minh

CHƯƠNG 2. QUÁ TẢI TOÁN TỬ

29/07/2023

ĐẠI HỌC CHÍNH QUY

Nội dung bài học

- ✓ **Quá tải toán tử (Operator Overloading)**

Các toán tử trong C#

- ✓ Toán học: +, -, *, /, %
- ✓ Cộng trừ một ngôi: ++, --
- ✓ Quan hệ so sánh: ==, !=, >, <, >=, <=
- ✓ Chuyển kiểu: ()
- ✓ Toán tử logic: &, |, ^, &&, ||
- ✓ Toán tử gán: +=, -=, *=, /=, %=

Quá tải toán tử trong C#

- ✓ Trong C#, toán tử là **một phương thức tĩnh** dùng để chuyển tải một phép toán nào đó trên các đối tượng.
- ✓ **Mục đích quá tải toán tử:** Viết mã chương trình gọn gàng, tự nhiên hơn **thay vì phải gọi phương thức** (nhưng bản chất vẫn là gọi phương thức)

Ví dụ: Class Phanso, thực thi chức năng cộng 2 phân số:

```
PS_Tong = PS_1.Cong(PS_2);
```

Nhưng điều ta muốn là:

```
PS_Tong = PS_1 + PS_2;
```

Quá tải toán tử trong C#

- ✓ **Quá tải toán tử** là định nghĩa lại chức năng của một toán tử trên những đối tượng lớp do người dùng định nghĩa.
- ✓ Khai báo giống như khai báo một phương thức (không có tên phương thức nhưng có danh sách tham số và kiểu trả về).

Cú pháp khai báo một toán tử

```
public static <KiểuTrảVề> operator<T> ([DSTS])
{
      //các câu lệnh trong thân toán tử
}
```

Trong đó:

- T: là toán tử.
- Bắt buộc phải khai báo **public static.**

Một số lưu ý

Các toán tử có thể quá tải:

- ✓ Toán học: +, -, *, /, %
- ✓ Cộng trừ một ngôi: ++, --
- ✓ Quan hệ so sánh: ==, !=, >, <, >=, <=
- ✓ Ép kiểu: ()

Một số lưu ý

- ✓ Tham số của toán tử phải là **tham trị** (không dùng các từ khóa ref, out).
- ✓ **Không được quá tải toán tử:** = (gán), &&, || (and, or logic), ?: (điều kiện), checked, unchecked, new, typeof, as, is.
- ✓ **[]**: không được xem là một toán tử.
- ✓ Khi quá tải các toán tử dạng: +, -, *, / , % thì các toán tử +=, -=, *=, /= , %= cũng tự động được quá tải.

Hạn chế

- ✓ **Tôn trọng ý nghĩa gốc của toán tử**
- ✓ Không thể tạo một toán tử mới hoặc kết hợp các toán tử có sẵn theo kiểu mà trước đó chưa định nghĩa.
- ✓ Không thể thay đổi thứ tự ưu tiên của các toán tử.
- ✓ Không thể định nghĩa lại một định nghĩa có sẵn của một toán tử
  - Ví dụ: Không thể thay đổi định nghĩa có sẵn của phép "+" đối với 2 kiểu số nguyên.

Ví dụ

Xây dựng lớp phân số và quá tải các phép toán trên phân số (+, –, *, /)

Ví dụ

Quá tải toán tử + phân số:

```
class PhanSo
{
    …
    public static PhanSo operator+ (PhanSo PS1,PhanSo PS2)
    {
        PhanSo ps = new PhanSo();
        ps.Mau = PS1.Mau * PS2.Mau ;
        ps.Tu = PS2.Mau * PS1.Tu + PS1.Mau * PS2.Tu;
        return ps;
    }
}
```


Ví dụ

Gọi thực thi:

```
static void Main(string[] args)
{
      PhanSo ps1 = new PhanSo(2, 3);
      PhanSo ps2 = new PhanSo(7, 5);
      PhanSo ps3 = ps1 + ps2;
      ps3.InPhanSo();
      Console.ReadKey();
}
```

Ví dụ

Trên lớp phân số, quá tải toán tử == (kiểm tra sự bằng nhau của 2 phân số)

```
class PhanSo
{
    …
    public static bool operator == (PhanSo PS1,PhanSo PS2)
    {
        if(PS1.GiaTriThuc == PS2.GiaTriThuc)
                return true;
        return false;
    }
}
```


Một số lưu ý

- ✓ Khi quá tải toán tử thì nên quá tải theo cặp đối ngẫu. Chẳng hạn, khi quá tải toán tử == thì quá tải thêm toán tử !=…

```
class PhanSo
{
    …
    public static bool operator != (PhanSo PS1,PhanSo PS2)
    {
        return !(PS1 == PS2);
    }
}
```

Một số lưu ý

- ✓ Khi quá tải toán tử ==, != thì nên phát triển thêm phương thức **`Equals()`** (phương thức ảo được cung cấp bởi lớp object)

```
class PhanSo
{
    …
    public static bool Equals (Object o)
    {
        if !(o is PhanSo)    return false;
        return this == (PhanSo)o;
    }
}
```


Ép kiểu

- ✓ **Ép kiểu** *là biến đổi dữ liệu thuộc kiểu dữ liệu này thành kiểu dữ liệu khác.*
- ✓ **Tại sao phải ép kiểu?**
  - Để chuyển dữ liệu sang một kiểu dữ liệu mong muốn phục vụ cho việc thao tác xử lý.
  - Đưa dữ liệu về định dạng mà mình mong muốn (ví dụ chuyển kiểu ngày tháng bên Mỹ sang dạng ngày tháng bên Việt Nam).

Các dạng Ép kiểu

- **Implicit:** ép kiểu ngầm định (không chỉ định), không mất dữ liệu, chuyển đổi này được thực hiện bởi trình biên dịch.

Ví dụ: ép từ kiểu số nguyên (kiểu dữ liệu nhỏ) sang kiểu số thực (kiểu dữ liệu lớn).

- **Explitcit:** ép kiểu tường minh (**có chỉ định**), có mất mát dữ liệu.

  Cú pháp: **(<kiểu dữ liệu>) <biến cần ép kiểu>**

Ví dụ: ép từ kiểu số thực sang số nguyên.

Các dạng Ép kiểu

✓ Sử dụng phương thức, lớp hỗ trợ sẵn để chuyển kiểu:

- Dùng phương thức **Parse(), TryParse().**
- Dùng lớp hỗ trợ sẵn (**Convert**).

Các dạng Ép kiểu

Phương thức **Parse()**

- Cú pháp:

<kiểu dữ liệu>.Parse(<dữ liệu cần chuyển đổi>);

Ví dụ:

```
string stringValue = "10";
int intValue = int.Parse(stringValue);
//Chuyển chuỗi stringValue sang kiểu int và lưu giá trị vào biến intValue
//Kết quả intValue = 10
```


Các dạng Ép kiểu

Phương thức **TryParse()**

- Cú pháp:

<kiểu dữ liệu>.TryParse(<dữ liệu cần chuyển đổi>, out <biến chứa kết quả>);

Ví dụ:

```
int Result; bool isSuccess;
string Data1 = "10";
isS = int.TryParse(Data1, out Result);
Console.Write(isS == true ? "Success!" : "Failed!");
                                                    //Success!
Console.WriteLine("Result = " + Result); //Result = 10
```

Các dạng Ép kiểu

Phương thức **TryParse()**

- Cú pháp:

<kiểu dữ liệu>.TryParse(<dữ liệu cần chuyển đổi>, out <biến chứa kết quả>);

Ví dụ:

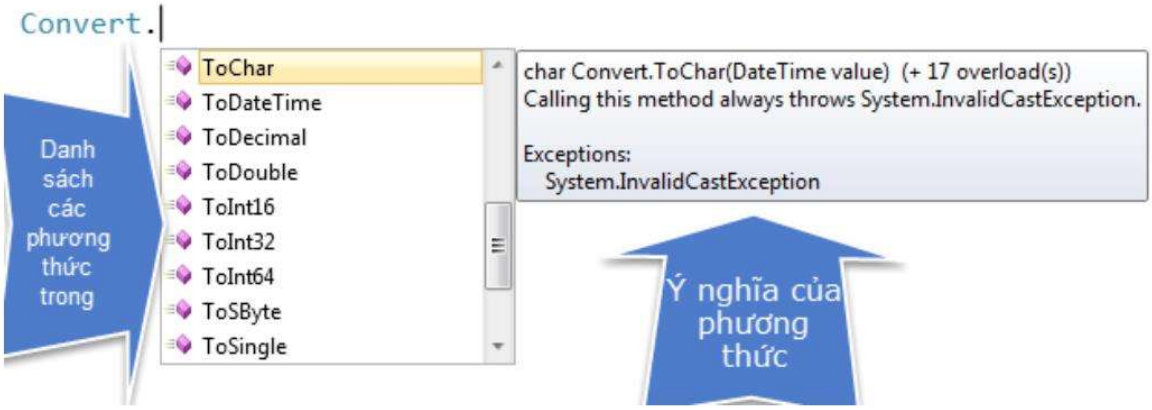
```
int Result; bool isSuccess;
string Data1 = "10DHTH";
isS = int.TryParse(Data1, out Result);
Console.Write(isS == true ? "Success!" : "Failed!");
                                                    //Failed!
Console.WriteLine("Result = " + Result); //Result = 0
```


Các dạng Ép kiểu

Lớp hỗ trợ sẵn **Convert**

- Là lớp tiện ích được C# hỗ trợ sẵn, cung cấp nhiều phương thức chuyển đổi kiểu dữ liệu.
- Trường hợp tham số truyền vào sai định dạng hoặc vượt quá giới hạn thì chương trình sẽ báo lỗi như phương thức **Parse().**

Các dạng Ép kiểu



Các dạng Ép kiểu

Một số phương thức chuyển kiểu mà **Convert** có:

Tên phương thức	Ý nghĩa
ToBoolean	Chuyển đổi một kiểu thành một bool (true, false) nếu có thể
ToByte	Chuyển đổi một kiểu thành một byte
ToChar	Chuyển đổi một kiểu thành một char nếu có thể
ToDateTime	Chuyển đổi một kiểu thành các cấu trúc date-time
ToDecimal	Chuyển đổi một kiểu thành một kiểu thập phân
ToDouble	Chuyển đổi một kiểu thành kiểu double
ToInt32	Chuyển đổi một kiểu thành kiểu int
ToString	Chuyển đổi một kiểu thành kiểu string

Người dùng định nghĩa toán tử ép kiểu

```
public static implicit|explicit operator
                              <KiểuTrảVề>(DataType V)
{
      //các câu lệnh trong thân toán tử
}
```

Trong đó:

- V: là biến cần ép sang kiểu <Kiểu trả về>.

Ví dụ

```
//Chuyen doi ngam dinh tu so nguyen sang phan so
public static implicit operator PhanSo(int theInt)
{
      return new PhanSo(theInt);
}
```

Gọi thực thi:

```
PhanSo ps = new PhanSo();
ps1 = 9;
ps1.XuatPS();
```

Ví dụ

```
//Chuyen doi tuong minh phan so sang so nguyen
public static explicit operator int(PhanSo PS)
{
      return PS.Tu/PS.Mau;
}
```

Gọi thực thi:

```
int e = (int)ps1;
Console.WriteLine("e = {0}", e);
```


Bài tập

Bài tập 1. Xây dựng lớp phân số và quá tải các phép toán trên phân số (==, !=, >, <)

Bài tập 2. Xây dựng class NgayThang lưu trữ ngày, tháng, năm với các toán tử: Cộng, Trừ, So sánh.