

LẬP TRÌNH .NET

Bài 1. Tổng quan về Microsoft .NET và ngôn ngữ C# (tiếp)

Căn bản về ngôn ngữ C#



MỤC TIÊU BÀI HỌC

- Bài học cung cấp các kiến thức tổng quan về Microsoft .NET và ngôn ngữ C#, đồng thời trang bị các kiến thức và kỹ năng sử dụng hằng, biến, kiểu dữ liệu và các cấu trúc điều khiển cơ bản trong C#.
- Sau khi học xong bài này sinh viên có khả năng:
 - Giải thích được cấu trúc của Microsoft .NET.
 - Trình bày được các đặc điểm của ngôn ngữ C#
 - Sử dụng được hằng, biến, kiểu dữ liệu và các cấu trúc điều khiển cơ bản



NỘI DUNG BÀI HỌC

- Câu lệnh (Statement)
- Định danh (Identifier)
- Từ khóa (Keyword)
- Biến
- Hằng
- Kiểu dữ liệu
- Biểu thức
- Các cấu trúc rẽ nhánh
- Các cấu trúc lặp



CÂU LỆNH (STATEMENT)

- Là hành động chúng ta yêu cầu chương trình thực hiện
- Đó có thể là khai báo biến, gán giá trị, gọi phương thức, xử lý điều kiện...
- Câu lệnh được kết thúc bằng dấu chấm phẩy (;)
- Ví dụ:

```
Console.WriteLine("Hello World!");
```



KHỐI LỆNH (CODE BLOCK)

- Một chuỗi câu lệnh có thể được nhóm lại với nhau tạo thành khối lệnh
- Một khối lệnh được đặt chung trong cặp dấu { }
- Các khối lệnh có thể lồng nhau
- Ví dụ:

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    string s = Console.ReadLine();
    Console.WriteLine(s);
    Console.ReadLine();
}
```



CHÚ THÍCH (COMMENTS)

- Chú thích (Comment): là một hoặc nhiều dòng văn bản, được chèn vào mã nguồn chương trình, nhằm làm cho mã nguồn trở nên dễ hiểu hơn với người đọc.
- Chú thích được bỏ qua bởi trình biên dịch.
- Có 2 cách viết chú thích:
 - Chú thích trên 1 dòng
 - Chú thích trên nhiều dòng



CHÚ THÍCH (COMMENTS)

- Chú thích trên một dòng:
`//Hiển thị ra màn hình console chuỗi: Welcome to CSharp`
- Chú thích trên nhiều dòng:

```
/*
Project: MyFirstCSharpPrj
Description: displays a Welcome message
*/
```



TÙ KHÓA (KEYWORD)

- Là từ khóa được giành riêng cho ngôn ngữ .NET
- Có màu xanh da trời trong môi trường Visual Studio (mặc định)
- Ví dụ:

using, class, namespace, int . . .



BIẾN (VARIABLE)

- Là vùng nhớ được đặt tên, chứa giá trị có thể thay đổi được khi chương trình thực thi
 - Đặt tên biến theo quy tắc của identifier, rõ ràng và gợi nhớ
 - Phải khai báo biến trước khi sử dụng
 - Dùng tên để truy xuất và truy nhập biến



BIẾN (VARIABLE)

- Cú pháp khai báo biến:

kiểu_dữ_liệu tên_biến [=<giá trị>];

- Ví dụ:

```
string fullName= "Tran Van A";
//hoặc
string fullName;
fullName="Tran Van A";
```



HẰNG

- Tương tự như biến nhưng giá trị không thay đổi khi chương trình thực thi

- Cú pháp khai báo hằng

const kiểu_dữ_liệu tên_hằng = giá_trị;

- Ví dụ:

```
const float PI = 3.14;
```



KIỂU DỮ LIỆU

C# chia kiểu DL thành 2 tập hợp kiểu DL chính:

- Kiểu xây dựng sẵn (built-in): do ngôn ngữ cung cấp cho người lập trình.
- Kiểu do người dùng định nghĩa (user-defined)



KIỂU DỮ LIỆU

Từ khóa trong C#	Số byte	Kiểu trong .NET	Mô tả
byte	1	Byte	Số nguyên dương từ 0 đến 255
int	4	Int32	Số nguyên từ -2.147.438.648 đến +2.147.438.647



KIỂU DỮ LIỆU

Từ khóa trong C#	Số byte	Kiểu trong .NET	Mô tả
float	4	Single	Số thực với độ chính xác tới 7 chữ số phần thập phân
double	8	Double	Số thực với độ chính xác tới 14 chữ số phần thập phân
decimal	16	Decimal	Số thực với độ chính xác lênh tới 28 chữ số phần thập phân



KIỂU DỮ LIỆU

Từ khóa trong C#	Số byte	Kiểu trong .NET	Mô tả
char	2	Char	Biểu diễn 1 ký tự Unicode
bool	1	Boolean	Biểu diễn giá trị true hoặc false
string		String	Chuỗi các ký tự, mỗi ký tự 2 byte

Có nhiều kiểu dữ liệu xây dựng sẵn khác trong C#, hãy tự tìm hiểu thêm về chúng!



CHUYỂN ĐỔI KIỂU DỮ LIỆU

- Sử dụng phương thức được định nghĩa sẵn trong mọi cấu trúc dữ liệu

Phương thức	Mô tả
ToString([format])	Chuyển đổi giá trị sang chuỗi tương ứng. Nếu bỏ qua việc định dạng chuỗi thì chuỗi kết quả sẽ không được định dạng
Parse(string)	Phương thức tĩnh chuyển đổi một chuỗi thành kiểu dữ liệu tương ứng



CHUYỂN ĐỔI KIỂU DỮ LIỆU

- Sử dụng phương thức tĩnh của lớp Convert

Phương thức	Mô tả
ToDecimal(value)	Chuyển đổi giá trị sang kiểu decimal
ToDouble(value)	Chuyển đổi giá trị sang kiểu double
ToInt32(value)	Chuyển đổi giá trị sang kiểu int
ToBool(value)	Chuyển đổi giá trị sang kiểu bool
ToString(value)	Chuyển đổi giá trị sang kiểu string
ToChar(value)	Chuyển đổi giá trị sang kiểu char



BIỂU THỨC

- Biểu thức bao gồm toán hạng và toán tử hoặc các phép logic
- Tuân theo thứ tự ưu tiên:
 - Trong ngoặc
 - Lũy thừa
 - Số âm
 - Nhân | chia
 - Chia lấy nguyên
 - Chia lấy dư



BIỂU THỨC

Toán tử

- **Arithmetic (toán học):** + - * / %
- **Assignment (gán):** = += -= *= /= %=
- **Unary (một ngôi):** ++ --
- **Comparison (so sánh):** kết quả là kiểu boolean sau khi đã so sánh
 - < <= == != > >=
- **Logical (luận lý)** - đánh giá biểu thức và trả về kiểu Boolean:
 - && (và) ! (phủ định) || (hoặc)



BIỂU THỨC

Toán tử gán

Toán tử	Ví dụ	Ý nghĩa
=	$x = 5$	gán giá trị 5 cho biến x
+=	$x += y$	tương tự: $x = x + y$
-=	$x -= y$	tương tự: $x = x - y$
*=	$x *= y$	tương tự: $x = x * y$
/=	$x /= y$	tương tự: $x = x / y$
%=	$x %= y$	tương tự: $x = x \% y$



BIỂU THỨC

Toán tử một ngôi

Toán tử	Ý nghĩa	Ví dụ
<code>++</code>	tăng giá trị của toán hạng lên 1	$y = ++x \rightarrow$ tăng x lên 1 rồi gán giá trị của x cho y $y = x++ \rightarrow$ gán y bằng giá trị của x rồi tăng x lên 1
<code>- -</code>	giảm giá trị của toán hạng đi 1	$y = --x \rightarrow$ giảm x rồi gán giá trị của x cho y $y = x-- \rightarrow$ gán y bằng giá trị của x rồi giảm x đi



LỆNH XUẤT DỮ LIỆU

Console.WriteLine(...);

Hoặc **Console.WriteLine(...);**

Ví dụ:

```
string str="Hà Giang";
```

```
Console.WriteLine("Chào bạn:" + str);
```



LỆNH NHẬP DỮ LIỆU

+ Nhập dữ liệu cho biến chuỗi:

biến=Console.ReadLine();

Ví dụ: string s=Console.ReadLine();

+ Nhập dữ liệu cho biến không phải kiểu chuỗi

C1: **biến=<Kiểu_DL_C#>.Parse(Console.ReadLine());**

C2: **biến=Convert.ToInt<Kiểu_DL_.NET>(Console.ReadLine());**



LỆNH NHẬP DỮ LIỆU

Ví dụ:

```
int a=int.Parse(Console.ReadLine());
```

Hoặc int a=Convert.ToInt32(Console.ReadLine());

```
double b=double.Parse(Console.ReadLine());
```

Hoặc double b= Convert.ToDouble(Console.ReadLine());



LỆNH XUẤT/NHẬP DỮ LIỆU

Ví dụ: Viết chương trình nhập vào: mã số(int), họ tên(string), lương(double). In ra màn hình các giá trị vừa nhập.

```
static void Main(string[] args)
{
    Console.Write("Vào mã số:");
    int ms = int.Parse(Console.ReadLine());
    Console.Write("Vào họ tên:");
    string ht =Console.ReadLine();
    Console.Write("Vào lương:");
    double l =double.Parse(Console.ReadLine());
    Console.WriteLine("Mã số: {0}; Họ tên:{1}; Lương:{2}", ms, ht, l);
    Console.ReadLine();
}
```



CẤU TRÚC IF

Câu lệnh if đơn giản

- Cú pháp

```
if (biểu thức điều kiện)
{
    //Khối lệnh A
}
```

- Ý nghĩa:

Nếu *biểu thức điều kiện* đúng thì thực hiện **Khối lệnh A**



CẤU TRÚC IF

Câu lệnh if ...else

Cú pháp

```
if (biểu thức điều kiện)
{
    //Khối lệnh A
}
else
{
    //Khối lệnh B
}
```

Ý nghĩa:

Nếu *biểu thức điều kiện* đúng thì
thực hiện Khối lệnh A
Ngược lại, *biểu thức điều kiện* sai
thực hiện Khối lệnh B



CẤU TRÚC IF

Câu lệnh if ...else if

Cú pháp

```
if (biểu thức điều kiện 1)
{
    //Khối lệnh 1
}
else if (biểu thức điều kiện 2)
{
    //Khối lệnh 2
}
.....
else
{
    //Khối lệnh n
}
```

Ý nghĩa:

Nếu *biểu thức điều kiện 1* đúng thì
thực hiện Khối lệnh 1

Ngược lại, *biểu thức điều kiện 2* đúng thì
thực hiện Khối lệnh 2

.....

Ngược lại tất cả các điều kiện trên thì
thực hiện Khối lệnh n



CẤU TRÚC IF

- Chú ý:

- Nếu khối lệnh chỉ có 1 câu lệnh thì không cần dùng cặp ngoặc nhọn { }
- Nếu có một chuỗi các điều kiện xử lý liên tục, có thể sử dụng nhiều if lồng nhau.



CẤU TRÚC SWITCH

Cú pháp

```
switch(biểu thức điều khiển)
{
    case giá trị 1:
        //Tập lệnh 1
        break;
    case giá trị 2:
        //Tập lệnh 2
        break;
    ...
    default:
        //Tập lệnh n
        break;
}
```

Ý nghĩa

- **case**: liệt kê các trường hợp cần xét.
- **giá trị i**: là các giá trị hằng cần so sánh với biểu thức điều khiển.
- Nếu biểu thức điều khiển bằng giá trị i thì **tập lệnh i** được thực hiện.
- Nếu các case không thỏa thì thực hiện **tập lệnh default** (nếu có).



CẤU TRÚC SWITCH

Chú ý:

- Kết thúc phát biểu case phải có break.
- Chỉ sử dụng switch cho các kiểu dữ liệu nguyên thủy như: int, string, char, bool... (biến thuộc kiểu dữ liệu hữu hạn, đếm được) những kiểu khác nên sử dụng câu lệnh if.
- Các giá trị phải là một hằng số như: 10, “Hà nội”... Nếu cần tính toán dữ liệu so sánh khi runtime thì nên dùng câu lệnh if.



CẤU TRÚC SWITCH

Chú ý:

- Các giá trị của các phát biểu case phải khác nhau, không có 2 case có cùng giá trị
- Nếu muốn nhiều case cùng thực hiện một tập lệnh thì viết như sau:

...

case giá trị 1:

case giá trị 2:

...

//Tập lệnh chung



CẤU TRÚC WHILE

Cú pháp

```
while (bíểu_thúc_logic)
{
    // Khối lệnh
    ...
}
```

Ý nghĩa:

Kiểm tra *bíểu_thúc_logic*, nếu thỏa mãn thì thực hiện *khối lệnh*. Tiếp tục thực hiện các lệnh cho đến khi *bíểu_thúc_logic* là false



CẤU TRÚC WHILE

```
int n = 1;
while (n < 6)
{
    Console.WriteLine("Current value of n is {0}", n);
    n++;
}
/*
Output:
Current value of n is 1
Current value of n is 2
Current value of n is 3
Current value of n is 4
Current value of n is 5
*/
```



CẤU TRÚC DO . . . WHILE

Cú pháp

```
do
{
    // khối lệnh
    ...
} while (biểu_thức_logic);
```

Ý nghĩa:

Khối lệnh trong vòng lặp sẽ được thực thi trước, sau đó kiểm tra *biểu_thức_logic*, nếu thỏa mãn thì khối lệnh tiếp tục được thực thi, nếu biểu thức logic không thỏa mãn thì vòng lặp sẽ dừng lại



CÂU TRÚC DO . . . WHILE

```
    . . .
int x = 0;
do
{
    Console.WriteLine(x);
    x++;
} while (x < 5);
/*
    Output:
    0
    1
    2
    3
    4
*/
```



CẤU TRÚC FOR

Sử dụng khi đã biết trước số lần lặp.

Cú pháp

```
for (khởi_tạo_biến_lặp; điều_kiện_lặp; cập_nhật_biến_lặp)
{
    //khối lệnh
}
```

Trong đó:

- **khởi_tạo_biến_lặp**: khai báo và gán giá trị ban đầu cho một biến đếm.
- **điều_kiện_lặp**: xác định điều kiện dừng của vòng lặp.
- **cập_nhật_biến_lặp**: tăng hoặc giảm giá trị của biến đếm sau mỗi lần lặp



CẤU TRÚC FOR

```
    . . .
for (int i = 1; i <= 5; i++)
{
    Console.WriteLine(i);
}
/*  
Output:  
1  
2  
3  
4  
5  
*/
```



CÂU TRÚC FOREACH

Cú pháp

foreach (kiểu biến in tập_hợp)

{

//khối lệnh

}

Ý nghĩa:

Lệnh foreach thực hiện duyệt qua từng phần tử trong **tập_hợp** để thực hiện các lệnh trong **khối lệnh**.

- **biến**: biến chạy để duyệt qua từng phần tử của **tập_hợp**.
- **Kiểu của biến** phải phù hợp với kiểu dữ liệu của các phần tử trong **tập_hợp**.



CÂU TRÚC FOREACH

```
....  
string[ ] values = {"Toi", "yeu", "Ha", "Noi"};  
foreach (string s in values)  
{  
    Console.WriteLine(s);  
}  
/*  
Output:  
Toi  
yeu  
Ha  
Noi  
*/
```



SỬ DỤNG BREAK & CONTINUE TRONG VÒNG LẶP

- Lệnh *break* thực hiện việc dừng vòng lặp.
- Khi chương trình đang chạy mà gặp lệnh *break* thì chương trình sẽ lập tức chấm dứt vòng lặp cho dù điều kiện của vòng lặp vẫn cho phép chạy tiếp.
- Trong trường hợp có nhiều vòng lặp lồng nhau, chương trình sẽ chấm dứt vòng lặp gần với *break* nhất.



SỬ DỤNG BREAK & CONTINUE TRONG VÒNG LẶP

```
//In ra số nguyên lớn nhất trong khoảng từ 1 đến 100 chia hết cho n
//(0<n<=100)

int i, max = 0;
int n = int.Parse(Console.ReadLine());
for (i = 100; i >= 1; i--)
{
    if (i % n == 0)
    {
        max = i;
        break; //thoat khoi vong lap
    }
}
Console.WriteLine("So lon nhat chia het cho {0} la {1}", n, max);
```



SỬ DỤNG BREAK & CONTINUE TRONG VÒNG LẶP

- Trong khi thực hiện vòng lặp, người lập trình đôi khi cần thực hiện việc bỏ qua một số dòng lệnh để tiếp tục thực hiện việc lặp cho lần tiếp theo.
- Lệnh *continue* thực hiện việc chuyển sang lần lặp tiếp theo và bỏ qua các lệnh nằm trong vòng lặp nhưng nằm phía sau nó.



SỬ DỤNG BREAK & CONTINUE TRONG VÒNG LẶP

```
//In ra tổng các số nguyên chẵn từ 1 đến 100)

int tong = 0;
for (int i = 1; i <= 100; i++)
{
    if (i % 2 != 0)
        continue;
    tong += i;
}
Console.WriteLine("Tong cac so nguyen chan tu 1 den 100 la:" + tong);
/*
Output:
Tong cac so nguyen chan tu 1 den 100 la:2550
*/
```



CẢM ƠN ĐÃ CHÚ Ý THEO DÕI !

