

LẬP TRÌNH WINDOWS

BÀI 2: CÁC KHÁI NIỆM CƠ BẢN TRONG C#

Giảng viên: Lý Anh Tuấn

Email: luanla@wru.vn

Các khái niệm cơ bản trong C#

- Các kiểu dữ liệu
- Các từ khóa
- Các toán tử
- Biến và hằng
- Các cấu trúc điều khiển

Các kiểu dữ liệu trong C#

Kiểu C#	Số byte	Kiểu .NET	Mô tả
byte	1	Byte	Số nguyên dương không dấu từ 0 đến 255
char	2	Char	Ký tự Unicode
bool	1	Boolean	Giá trị logic true / false
sbyte	1	Sbyte	Số nguyên có dấu từ -128 đến 127
short	2	Int16	Số nguyên có dấu từ -32768 đến 32767
ushort	2	UInt16	Số nguyên dương không dấu từ 0 đến 65535
int	4	Int32	Số nguyên có dấu từ -2.147.483.647 đến 2.147.483.647
uint	4	UInt32	Số nguyên không dấu từ 0 đến 4.294.967.295
float	4	Single	Kiểu dấu chấm động, giá trị xấp xỉ từ -3.4E-38 đến 3.4E+38, với 7 chữ số có nghĩa
double	8	Double	Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ -1.7E-308 đến 1.7E+308, với 15, 16 chữ số có nghĩa
decimal	8	Decimal	Có độ chính xác đến 28 con số và giá trị thập phân, được dùng trong tính toán tài chính, kiểu này đòi hỏi phải có hậu tố “m” hay “M”
long	8	Int64	Kiểu số nguyên có dấu có giá trị trong khoảng -9.223.370.036.854.775.808 đến 9.223.372.036.854.775.807
ulong	8	UInt64	Số nguyên không dấu từ 0 đến 0xffffffffffffffff

Các từ khóa trong C#

abstract	<u>default</u>	<u>foreach</u>	<u>object</u>	<u>sizeof</u>	<u>unsafe</u>
as	<u>delegate</u>	<u>goto</u>	<u>operator</u>	<u>stackalloc</u>	<u>ushort</u>
base	<u>do</u>	<u>if</u>	<u>out</u>	<u>static</u>	<u>using</u>
bool	<u>double</u>	<u>implicit</u>	<u>override</u>	<u>string</u>	<u>virtual</u>
break	<u>else</u>	<u>in</u>	<u>params</u>	<u>struct</u>	<u>volatile</u>
byte	<u>enum</u>	<u>int</u>	<u>private</u>	<u>switch</u>	<u>void</u>
case	<u>event</u>	<u>interface</u>	<u>protected</u>	<u>this</u>	<u>while</u>
catch	<u>explicit</u>	<u>internal</u>	<u>public</u>	<u>throw</u>	
char	<u>extern</u>	<u>is</u>	<u>readonly</u>	<u>true</u>	
checked	<u>false</u>	<u>lock</u>	<u>ref</u>	<u>try</u>	
class	<u>finally</u>	<u>long</u>	<u>return</u>	<u>typeof</u>	
const	<u>fixed</u>	<u>namespace</u>	<u>sbyte</u>	<u>uint</u>	
continue	<u>float</u>	<u>new</u>	<u>sealed</u>	<u>ulong</u>	
decimal	<u>for</u>	<u>null</u>	<u>short</u>	<u>unchecked</u>	

Xuất các ký tự đặc biệt

Ký tự	Ý nghĩa
\'	Dấu nháy đơn
\"	Dấu nháy kép
\\	Dấu chéo
\0	Ký tự null
\a	Alert
\b	Backspace
\f	Sang trang form feed
\n	Dòng mới
\r	Đầu dòng
\t	Tab ngang
\v	Tab dọc

Các toán tử

Toán tử một ngôi	Ý nghĩa	Ví dụ
++	Tăng 1	$++a \approx a = a+1$ // tiền tố, tăng trước khi sd a $a++ \approx a = a+1$ // hậu tố, tăng sau khi sd a
--	Giảm 1	$--a \approx a = a-1$ //tiền tố, giảm trước khi sd a $a-- \approx a = a-1$ //hậu tố, giảm sau khi sd a
-	Lấy phủ định	-a lấy phủ định của số a

Các toán tử

Toán tử hai ngôi	Ý nghĩa
=	Toán tử gán
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia lấy phần nguyên
%	Phép chia lấy phần dư

Các toán tử

Toán tử tự gán (2 ngôi)	Ý nghĩa
<code>+=</code>	$a += b \leftrightarrow a = a + b$
<code>-=</code>	$a -= b \leftrightarrow a = a - b$
<code>*=</code>	$a *= b \leftrightarrow a = a * b$
<code>/=</code>	$a /= b \leftrightarrow a = a / b$
<code>%=</code>	$a \% = b \leftrightarrow a = a \% b$

Toán tử điều kiện (3 ngôi)

- Cú pháp:

Kết quả = (biểu thức kiểm tra) ? giá trị đúng : giá trị sai

- VD: **max = (a>b) ? a : b;**
- Tương đương câu lệnh *if ... else* như sau:

if(a>b)

max = a;

else

max = b;

Toán tử quan hệ

Toán tử	Mô tả	Ví dụ
<code>==</code>	So sánh bằng	<code>7 == 5</code> // trả về false <code>(b = 2) == 5</code> // trả về false
<code>!=</code>	Khác	<code>(3 != 2)</code> // trả về true
<code><</code>	Nhỏ hơn	<code>(5 < 5)</code> // trả về false
<code>></code>	Lớn hơn	<code>(3 > 2)</code> // trả về true
<code><=</code>	Nhỏ hơn hoặc bằng	<code>(6 <= 6)</code> // trả về true
<code>>=</code>	Lớn hơn hoặc bằng	<code>(6 >= 4 +2)</code> // trả về true

Toán tử logic

Toán tử	Chức năng
&&	Trả kết quả là true khi cả 2 toán hạng đều là true
 	Trả về kết quả là true khi chỉ một trong 2 toán hạng là true
!	Chuyển đổi giá trị từ true thành false và ngược lại

Chuyển đổi kiểu dữ liệu

- Chuyển đổi ngầm định

- Được thực hiện ngầm định
- Không bị mất thông tin
- Ví dụ:

```
short x = 20;
```

```
int y = x; //=> chuyển đổi ngầm định và không mất thông tin, vì mọi giá trị kiểu short đều thuộc về int
```

Chuyển đổi dữ liệu

- Chuyển đổi tường minh

- Gán ép một giá trị cho một biến thuộc kiểu dữ liệu khác: <tên biến> = (tên kiểu)<biến kiểu lớn hơn>;

- Ví dụ:

`short x; int y = 100; x = y;` //không thực hiện được vì kiểu của x < kiểu của y => việc chuyển ngầm định sẽ bị mất thông tin.

Như vậy, muốn phép gán này không bị lỗi thì phải viết như sau:

`x = (short)y;` //chuyển đổi tường minh/ép kiểu

Chuyển đổi dữ liệu

- Chuyển đổi tương minh
 - Sử dụng các lệnh chuyển kiểu trong lớp Convert thuộc namespace System
 - Ví dụ:

```
int a;
```

```
a = Convert.ToInt32(Console.ReadLine());
```

Bài tập

Viết chương trình nhập tên, năm sinh. Xuất ra lời chào tên vừa nhập và thông báo số tuổi của người đó dựa vào năm sinh

Biến và hằng

- Cú pháp khai báo biến:

<Kiểu_Dữ_Liệu> <tên_biến> [= <giá_trị>];

- Ví dụ:

- `int a;` //khai báo biến a kiểu số nguyên
- `int x = 10;` //khai báo biến x kiểu số nguyên và gán giá trị khởi tạo ban đầu cho x là 10

Biến và hằng

- Khởi tạo và gán giá trị cho biến

```
0 references
class MinhHoaC1
{
    0 references
    static void Main()
    {
        int myVar = 0;
        System.Console.WriteLine("Sau khi khoi tao: myVar ={0}", myVar);
        myVar = 10;
        System.Console.WriteLine("Sau khi gan: myVar ={0}", myVar);
    }
}
```

Biến và hằng

- Cú pháp khai báo hằng:

const <Kiểu_Dữ_Liệu> <tên_hằng> = <giá_trị>;

- Ví dụ:

- **const int DoSoi = 100;** //khai báo hằng kiểu nguyên tên là DoSoi có giá trị không thể thay đổi là 100

Kiểu liệt kê

- Được sử dụng để tạo một tập hằng có liên hệ ngữ nghĩa
- Ví dụ:

```
enum KichThuoc :uint
{
    Nho = 1,
    Vua = 2,
    Lon = 3,
}
```

Kiểu liệt kê

```
class MinhHoaC1
{
    // Khai báo kiểu liệt kê
    5 references
    enum NhiệtĐộNước
    {
        ĐộĐông = 0,
        ĐộNgười = 20,
        ĐộẤm = 40,
        ĐộNóng = 60,
        ĐộSôi = 100,
    }
    0 references
    static void Main()
    {
        System.Console.WriteLine("Nhiệt độ đông: {0}", NhiệtĐộNước.ĐộĐông);
        System.Console.WriteLine("Nhiệt độ người: {0}", NhiệtĐộNước.ĐộNgười);
        System.Console.WriteLine("Nhiệt độ ấm: {0}", NhiệtĐộNước.ĐộẤm);
        System.Console.WriteLine("Nhiệt độ nóng: {0}", NhiệtĐộNước.ĐộNóng);
        System.Console.WriteLine("Nhiệt độ sôi: {0}", NhiệtĐộNước.ĐộSôi);
    }
}
```

Các cấu trúc điều khiển

- Giống như trong C++
- Cấu trúc rẽ nhánh:
 - if
 - if....else
 - switch....case
 - Toán tử điều kiện

Câu lệnh rẽ nhánh if

- Cú pháp:

```
if (Biểu_thức_Boolean_1)  
    Câu_lệnh_1;
```

- Hoặc:

```
if (Biểu_thức_Boolean_1)  
{  
    Câu_lệnh_1;  
    Câu_lệnh_2;  
}
```

Các câu lệnh sẽ được thực hiện khi biểu thức kiểm tra có kết quả là đúng

Câu lệnh rẽ nhánh if...else

- Cú pháp

if (Biểu_thức_Boolean)

//Câu lệnh khi biểu thức Boolean đúng

else

//Câu lệnh khi biểu thức Boolean sai

Câu lệnh rẽ nhánh if...else


- Hoặc

```
if (Biểu_thức_Boolean){  
    Câu_lệnh_khi_đúng_1;  
    Câu_lệnh_khi_đúng_2;  
}  
else{  
    Câu_lệnh_khi_sai_1;  
    Câu_lệnh_khi_sai_2;  
}
```


Câu lệnh rẽ nhánh switch...case

- Việc lựa chọn nhánh nào để thực thi được dựa trên **biểu thức điều khiển**
- Biểu thức điều khiển cho một câu lệnh switch phải trả về một giá trị bool hoặc một bộ liệt kê các hằng số, hoặc một giá trị kiểu số nguyên hoặc một ký tự

```
switch (Bieu_thuc_dieu_khien)
{
    case Hang_so_1:
        Dãy_câu_lệnh_1
        break;
    case Hang_so_2:
        Dãy_câu_lệnh_2
        break;
    .
    .
    .
    case Hang_so_n:
        Dãy_câu_lệnh_n
        break;
    default:
        Dãy_câu_lệnh_mac_dinh
        break;
}
```



Câu lệnh rẽ nhánh switch...case

- Câu lệnh rẽ nhánh switch...case thường được sử dụng khi muốn thực hiện các khối lệnh khác nhau với mỗi lựa chọn khác nhau của một biểu thức hay giá trị của một biến
- Sau các câu lệnh của mỗi lựa chọn case nên dùng lệnh break để bỏ qua các lựa chọn case khác
- Sau các lựa chọn case nên có lựa chọn default để thông báo rằng chưa có lựa chọn case nào được chọn

Câu lệnh rẽ nhánh switch...case

- Có thể có trường hợp nhiều lựa chọn cùng thực hiện một công việc
- Ví dụ:

```
switch(thang)
{
    case 4:
    case 6:
    case 9:
    case 11:
        Console.WriteLine("Tháng {0} có 30 ngày", thang);
        break;
```

Các cấu trúc điều khiển

- Cấu trúc lặp
 - for
 - while
 - do... while
 - foreach

Cấu trúc vòng lặp for

- Được dùng khi biết trước số lần lặp
- Cú pháp:
`for` (khởi tạo biến chạy; kiểm tra biến chạy; thay đổi giá trị biến chạy)
{
 các câu lệnh;
}

Cấu trúc lặp for

- Ví dụ: tính tổng các số nguyên dương nhỏ hơn 10
 - Thực hiện gán biến $i = 0$
 - Lặp 10 lần, mỗi lần cộng i vào tổng rồi tăng i lên 1 đơn vị

```
int tong = 0;
for (int i = 1; i < 10; i++)
    tong = tong + i;
```

Hãy cho biết kết quả của vòng lặp sau?

```
int i;  
for(i=0; i<10; i-=2)  
    cout<<i<<" ";  
    cout<<endl;
```

Hãy cho biết kết quả của vòng lặp sau?

```
int i=0;  
for(;i<10; i++)  
    cout<<i<<" ";  
    cout<<endl;
```


Cấu trúc lặp while

- Được dùng khi không biết trước số lần lặp
- Cú pháp: `while`(biểu thức điều kiện)
 câu lệnh;
- Hoặc: `while`(biểu thức điều kiện)
 {
 câu lệnh 1;
 câu lệnh 2;

 }

Vòng lặp while

- Chú ý:
 - Vòng lặp dừng khi biểu thức kiểm tra điều kiện cho ra giá trị sai
 - Do đó khối lệnh trong vòng lặp while phải có lệnh làm thay đổi giá trị biến chạy sao cho biểu thức kiểm tra điều kiện trở thành sai.
 - Nếu không thì vòng lặp sẽ bị lặp vô hạn

Vòng lặp while

- Ví dụ:

- Lệnh $t = t/10$ là lệnh làm thay đổi giá trị biến chạy
- Sau mỗi lần thực hiện lệnh này, t sẽ giảm 10 lần, cho đến một lúc nào đó t sẽ bằng 0. Khi đó biểu thức kiểm tra $t > 0$ sẽ không còn đúng nữa và vòng lặp while sẽ dừng

```
while (t > 0)
{
    tong += t % 10;
    t = t / 10;
}
```

Cấu trúc lặp do...while

- Được dùng khi không biết trước số lần lặp nhưng cần thực hiện các câu lệnh ít nhất 1 lần trước khi kiểm tra điều kiện để lặp tiếp

- Cú pháp: `do`
 câu lệnh;
 `while`(biểu thức điều kiện);
- Hoặc: `do`
 {
 các câu lệnh;
 }
 `while`(biểu thức điều kiện);

Cấu trúc lặp do...while

- Chú ý: giống như vòng lặp while
 - Vòng lặp do...while sẽ dừng khi biểu thức kiểm tra điều kiện cho ra giá trị sai
 - Do đó khối lệnh trong vòng lặp do...while phải có lệnh làm thay đổi giá trị biến chạy sao cho biểu thức kiểm tra điều kiện dần bị sai.
 - Nếu không thì vòng lặp sẽ bị lặp vô hạn

Cấu trúc lặp do...while

- Ví dụ:

```
do
{
    Console.WriteLine("Nhap so n thoa man  $0 < n < 2000000$ : ");
    n = Convert.ToInt32(Console.ReadLine());
}
while (n <= 0 || n >= 2000000);
```

Đây chính là lệnh làm
cho vòng lặp dừng

Hãy cho biết kết quả của vòng lặp sau?

```
int n;  
do{  
    cout<<"Nhập N = ";  
    cin>>n;  
}  
while(n<0 && n>1000000);
```

Cấu trúc vòng lặp foreach

- Dùng khi duyệt tất cả các phần tử trong mảng (mảng đã được khởi tạo)
- Cú pháp: với **a** là một mảng các phần tử có giá trị, **x** là một biến cùng kiểu với các phần tử của mảng **a**

```
foreach(x in a)
{
    //công việc
}
```


Cấu trúc vòng lặp foreach

- Dùng khi duyệt tất cả các phần tử trong mảng
- Ví dụ:

```
//khai báo mảng a gồm 10 phần tử kiểu số nguyên
int[] a = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
//duyet từng phần tử của mảng a
foreach (int x in a)
    //viết các phần tử vừa duyệt lên cùng một dòng
    //mỗi phần tử cách nhau một dấu ; và một dấu cách
    Console.Write(x + "; ");
```

Kiến thức đã học trong bài

- Các kiểu dữ liệu
- Các từ khóa
- Các toán tử
- Hằng và biến
- Các cấu trúc điều khiển