BÀI 2:
LẬP TRÌNH C# CƠ BẢN

Thời gian: 180 phút

Giảng viên: PHAMPHUKHUONG

Email: phamphukhuong@dtu.edu.vn
Điện thoại:0905635421

DUYTAN edu.vn

NỘI DUNG

1. Những nội dung cơ bản
2. Các lệnh trong C#

NỘI DUNG CƠ BẢN

- Cấu trúc chương trình C#
- Kiểu dữ liệu, từ khoá, định danh biến, hằng…
- Chuyển đổi kiểu
- Console I/O
- Tham số ref, out, param

Video:
https://www.youtube.com/watch?v=2VkYpsKCzGg

Cấu trúc chương trình C#

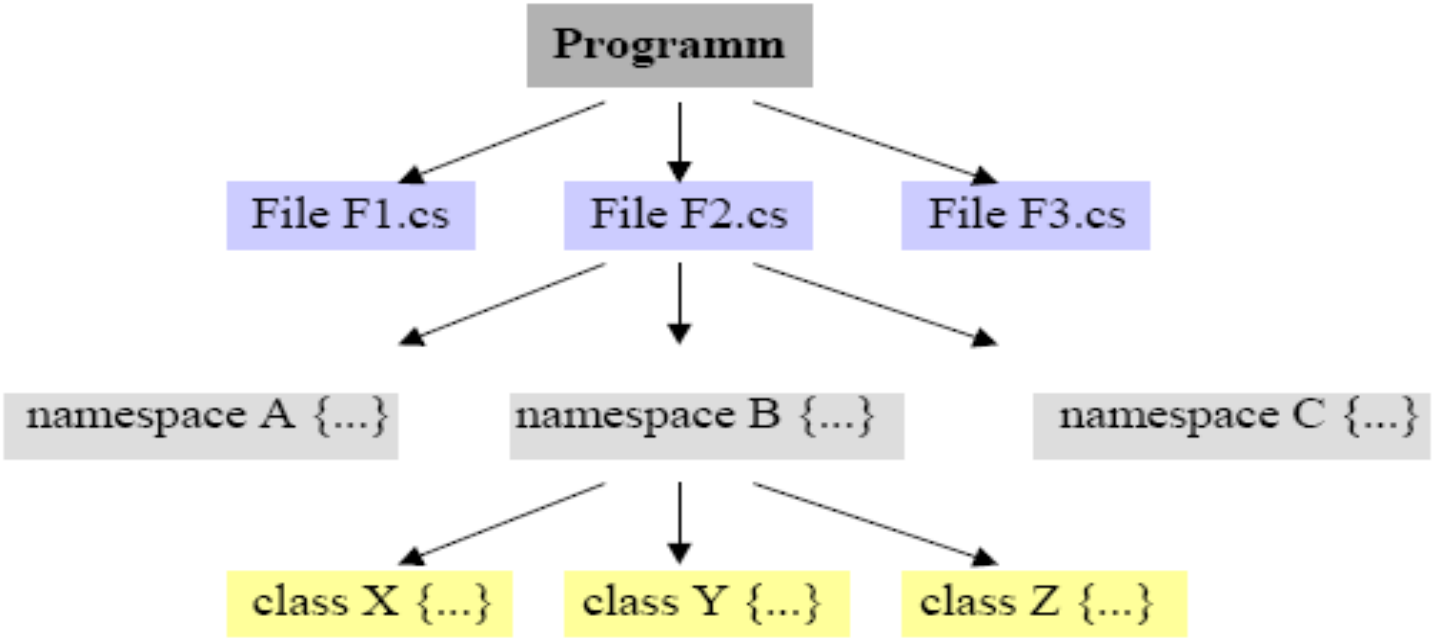
```
using <namespace sử dụng>

…
namespace <Tên Namespace>
{
        [Khóa truy xuất] class <Tên lớp>
        {
                public static void Main()
                {
                        …
                }
                // thành viên khác …
        }
        // lớp khác …
```

Cấu trúc chương trình C#

- **using**: làm code gọn hơn, ko cần phải dùng tên của namspace
  - using System.
- namespace của chương trình: ko bắt buộc
- **class**: tối thiểu có 1 lớp chứa hàm entry point Main của chương trình
- public static void Main(): hàm entry point của chương trình C#

Cấu trúc chương trình C#



- Nếu ko có namespace ⇒ namespace mặc định ko tên
- Namespace có thể chứa: struct, interface, delegate, enum
- Trường hợp đơn giản nhất: 1 lớp, 1 file cs và namespace mặc định

DUYTAN edu.vn

Kiểu dữ liệu

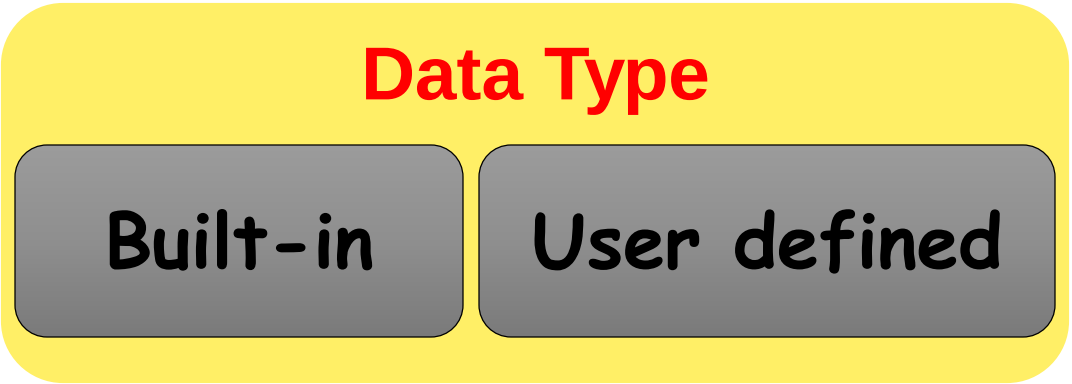
- Bao gồm
  - Lớp đối tượng — object
  - Ký tự, chuỗi — char, string
  - Số nguyên có dấu — sbyte, short, int, long
  - Số nguyên không dấu — byte, ushort, uint, ulong
  - Số thực — float, double, decimal
  - Kiểu logic — bool

Kiểu dữ liệu

- Là alias của lớp dữ liệu trong .NET
  - **s**tring System.**S**tring
  - **i**nt System.**I**nt32
  - **o**bject System.**O**bject

Kiểu dữ liệu

- Sử dụng kiểu dữ liệu
  - Định nghĩa trước (C#)
    - Built-in value type: int, long, string, object…
  - Chương trình định nghĩa (tham chiếu)
    - Class, struct, enum…
      - Person, Student, Employee…



Kiểu dữ liệu giá trị

Name	CTS Type	Size	Range
sbyte	System.SByte	8	-128..127
short	System.Int16	16	(-32768 .. 32767)
int	Sytem. Int32	32	$-2^{31}..2^{31}-1$
long	Sytem. Int64	64	$-2^{63}..2^{63}-1$
byte	System.SByte	8	0..255
ushort	System.UInt16	16	(0 .. 65535)
uint	System.UInt32	32	$0..2^{32}-1$
ulong	System.UInt64	64	$0..2^{64}-1$
float	System.Single	32	xấp xỉ từ 3,4E - 38 đến 3,4E+38
double	System.Double	64	1,7E-308 đến 1,7E+308
decimal	System.Decimal	128	Có độ chính xác đến 28 con số
bool	System.Boolean		Kiểu true/false
char	System.Char	16	Ký tự unicode

Kiểu giữ liệu tham chiếu

- object: Sytem.Object
  - Kiểu dữ liệu gốc, cha của tất cả các kiểu dữ liệu trong C#

```
object o = new object();
```

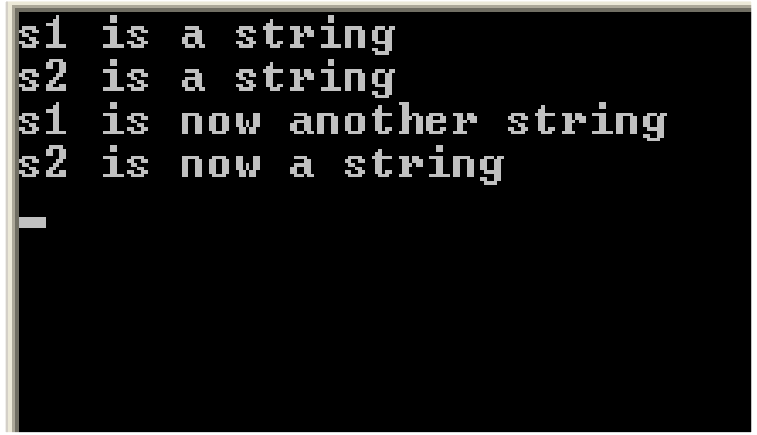
- string: Sytem.String
  - Chuỗi ký tự Unicode

```
string s1 = “Hutech”;
string s2 = “Hi”;
string s = s1 + s2;
```

Kiểu giữ liệu tham chiếu

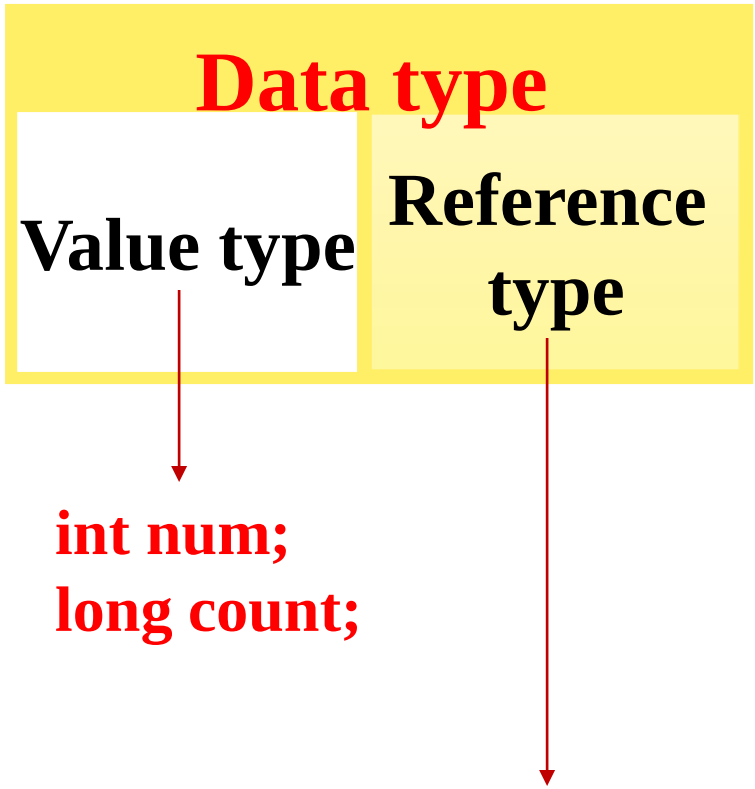
```
using System;

class StringExample
{
    public static int Main()
    {
        string s1 = "a string";
        string s2 = s1;
        Console.WriteLine("s1 is " + s1);
        Console.WriteLine("s2 is " + s2);
        s1 = "another string";
        Console.WriteLine("s1 is now " + s1);
        Console.WriteLine("s2 is now " + s2);
        return 0;
    }
}
```



Phân loại kiểu dữ liệu

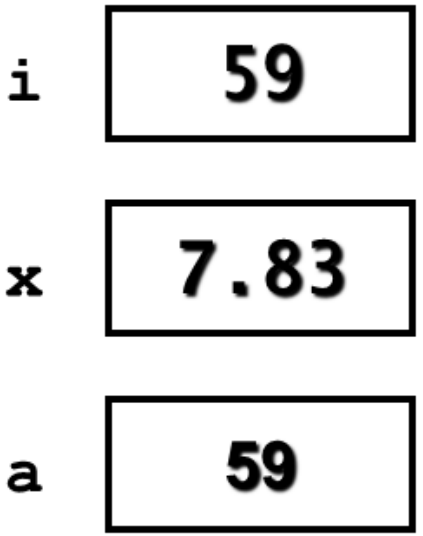
Phân loại theo cách thức lưu trữ dữ liệu



Value Type

- Chứa giá trị trực tiếp
- Không thể null
  - Phải chứa giá trị xác định
- Bao gồm
  - Primitive type
    - double, char, int, float
  - Enum
  - struct

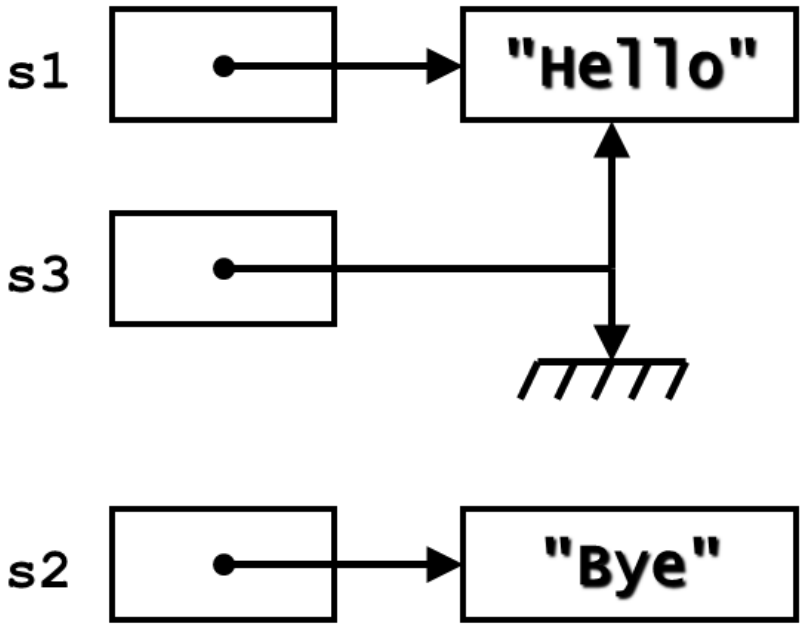
```
int i = 59;
double x = 7.83;
int a = i;
```



Reference type

- Chỉ tới nơi chứa dữ liệu
- Có thể **null**
  - **null**: không chỉ tới bất kỳ đâu
- Bao gồm
  - Lớp (class)
    - **string**, **object**
  - Giao diện (interface)
  - Mảng (array)
  - Đại diện (delegate)

```
string s1 = "Hello"
string s2 = "Bye"
string s3;
s3 = s1;
```



Value type vs. Reference type

Characteristic	Value type	Reference type
Variable hold	Value	Reference
Allocated	Stack	Heap
Default	Zero	Null
Parameter	Copy value	Copy reference

Định danh (Identifier)

- Định danh: những từ được đặt ra để đại diện cho mọi thứ dùng trong chương trình
  - Khi đặt định danh: nên có tính gợi nhớ
- Tạo ra định danh mới
  - **HelloWorld**, **Program**, **Perform**,…

→ phải khai báo trước khi sử dụng

- Dùng định danh có sẵn
  - **Console**, **WriteLine**, **ReadLine**,…

→ phải chỉ ra nơi chứa định danh (namespace)

Định danh (Identifier)

- Bao gồm chữ cái, chữ số, ký tự gạch dưới
- Không được bắt đầu bằng chữ số
  - **Chuong_Trinh**, **x25**, **z**, **_abc**, **XửLý** → hợp lệ
  - **2abc**, **Chuong-Trinh**, **Xu Ly**, **class** → không hợp lệ
- Case-sensitive
  - **ChuongTrinh** và **chuongtrinh** là khác nhau
- Các định danh được khai báo trong cùng phạm vi (scope) không được trùng nhau
- Phải khác với từ khóa (*có thể dùng "@" trước từ khóa làm identifier*)

Từ khóa (Keyword)

Các từ khóa trong C# 2005							
abstract	const	extern	in	operator	sbyte	throw	virtual
as	continue	false	int	out	sealed	true	void
base	decimal	finally	interface	override	set	try	volatile
bool	default	fixed	internal	params	short	typeof	where
break	delegate	float	is	partial	sizeof	uint	while
byte	do	for	lock	private	stackalloc	ulong	yield
case	double	foreach	long	protected	static	unchecked	
catch	else	get	namespace	public	string	unsafe	
char	enum	goto	new	readonly	struct	ushort	
checked	event	if	null	ref	switch	using	
class	explicit	implicit	object	return	this	value	

Hằng (Constant)

- Một hằng là một biến nhưng trị không thay đổi

  const int a = 100; // giá trị ko thể thay đổi
- Hằng bắt buộc phải được gán giá trị lúc khai báo
- Trị của hằng có thể được tính toán vào lúc biên dịch
- Hằng bao giờ cũng static

Hằng (Constant)

- Ưu điểm
  - Chương trình dễ đọc, khắc phục những con số "*magic number*" trong code.
  - Chương trình dễ sửa hơn.
  - Tránh lỗi dễ dàng hơn, trình biên dịch sẽ báo lỗi nếu gán lại giá trị cho hằng

Hằng (Constant)

- Minh họa sử dụng hằng

```
public void Test1()
{
    int[] a = new int[100];
    Random rand = new Random();
     for(int i=0; i < 100; i ++)
        a[i] = rand.Next(100);

    int k = 100;
    int j=0;
    while ( j < 100 && a[j] != k)
        j++;
}
```

Định nghĩa hằng

```
public void Test2()
{
    const int MAXLEN = 100;
    const int MAXNUM = 100;
    int[] a = new int[MAXLEN];
    Random rand = new Random();
    for (int i = 0; i < MAXLEN; i++)
        a[i] = rand.Next(MAXNUM);

    int k = 100;
    int j = 0;
    while (j < MAXLEN && a[j] != k )
        j++;
}
```

readonly

const: phải được gán giá trị khi khai báo

readonly: ko cần khởi tạo trước, khi gán giá trị thì sau đó ko thay đổi được

```
class MyClass
{
        const int x = 5;
        readonly int y = 25;
        const int xx;
        readonly int yy;
        public MyClass()
        {
            yy = 24;
        }
        public void MyMethod()
        {
            x = 7;
            yy = 30;
        }
```

Chưa được khởi gán

Ko được thay đổi

Biến (Variable)

- Biến là nơi lưu dữ liệu của chương trình
- Dữ liệu của biến
  - Nằm trong bộ nhớ vật lý (physical RAM)
  - Có thể thay đổi giá trị
- Phải khai báo trước khi dùng
  - Identifier: tên để đại diện cho biến
  - Data type: dạng lưu trữ dữ liệu của biến

Data type **identifier**

Biến (Variable)

- Phạm vi (scope)
  - Được xác định bởi cặp dấu **{** và **}**
  - Có thể chứa phạm vi nhỏ hơn
- Vị trí khai báo biến
  - Trong thân phương thức: biến cục bộ
  - Trong thân lớp: thuộc tính
- Biến trong C# chỉ có tác dụng trong phạm vi mà nó được khai báo

Ép kiểu (Type cast)

- Chuyển đổi kiểu dữ liệu (ép kiểu): chuyển giá trị từ kiểu này sang kiểu khác (có liên quan với nhau)
- Ví dụ
  - Chuyển từ **int** qua **float** và ngược lại
- Có hai loại
  - Ép kiểu ngầm định
  - Ép kiểu tường minh

Ép kiểu ngầm định

- Do C# tự thực hiện
- Không bị mất thông tin
- Không cần lập trình viên can thiệp
- Xảy ra khi nào
  - Ép từ kiểu nhỏ sang kiểu lớn

```
int i = 59;
float x = i;
```

  - Sau khi gán thì hoàn toàn không bị mất dữ liệu vì bất cứ giá trị nào của int cũng thuộc về float.

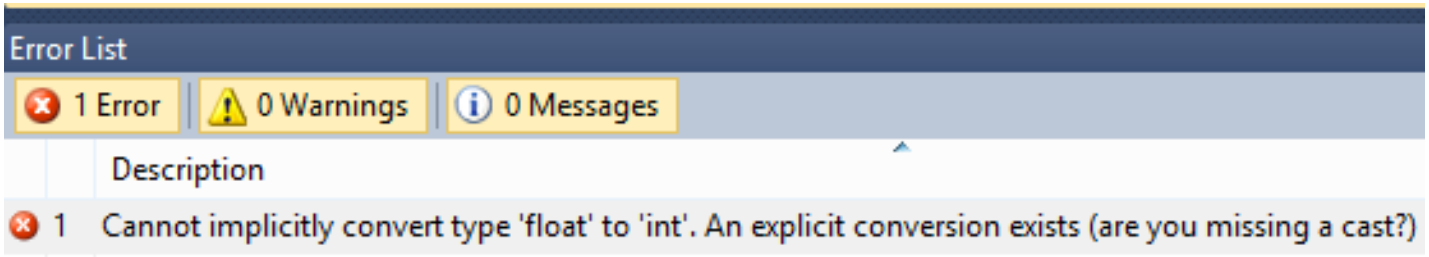
Ép kiểu ngầm định

From	To
sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long, ulong	float, double, decimal
float	double

Ép kiểu tường minh

– Ép từ kiểu lớn sang kiểu nhỏ

```
float i = 59;
int x = i;
```



– Buộc phải dùng lệnh gán tường minh

```
float i = 59;
int x = (int)i;
```

Lớp Convert

- Thường dùng khi cần chuyển đổi giữa các kiểu dữ liệu không liên quan với nhau

Convert.KiểuDữLiệu(*GiáTrịNguồn*)

❖ **Ví dụ chuyển đổi từ kiểu chuỗi sang số thực**

```
string s1 = "56.8";
string s2 = "95";
double x = Convert.ToDouble(s1); // x = 56.8
int i = Convert.ToInt32(s2);     // i = 95
byte j = Convert.ToByte(x);// j = 56, ít dùng
```

Phương thức Parse

- Thường dùng khi cần chuyển đổi giữa các kiểu dữ liệu không liên quan với nhau

KiểuDữLiệu.Parse(*GiáTrịNguồn*)

❖ **Ví dụ chuyển đổi từ kiểu chuỗi sang số thực**

```
string s1 = "56.8";
string s2 = "95";
double x = double.Parse(s1);     // x = 56.8
int i = int.Parse(s2);     // i = 95
```

Câu lệnh nhập xuất

- Để đọc văn bản từ cửa sổ console
  - Console.ReadLine() giá trị trả về là string

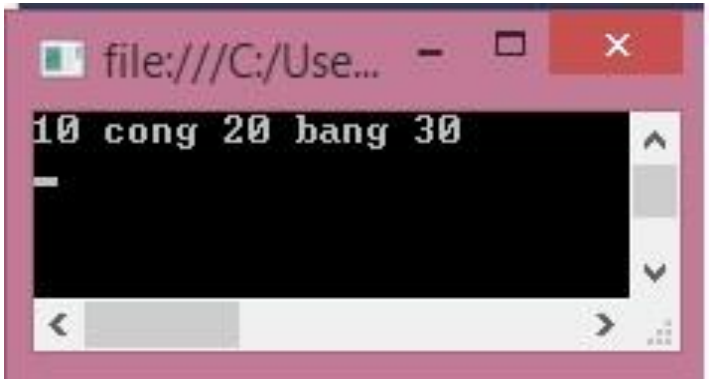
```
string s = Console.ReadLine();
Console.WriteLine(s);
```

- Để xuất chuỗi ký tự thì ta dùng
  - Console.Write() / Console.WriteLine()

Câu lệnh nhập xuất

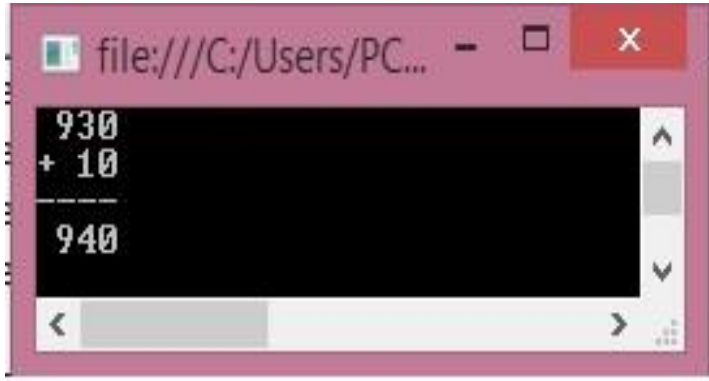
- Để xuất chuỗi dùng Console.WriteLine()

```
int a = 10;
int b = 20;
Console.WriteLine("{0} cong {1} bang {2}", a, b, a + b);
```



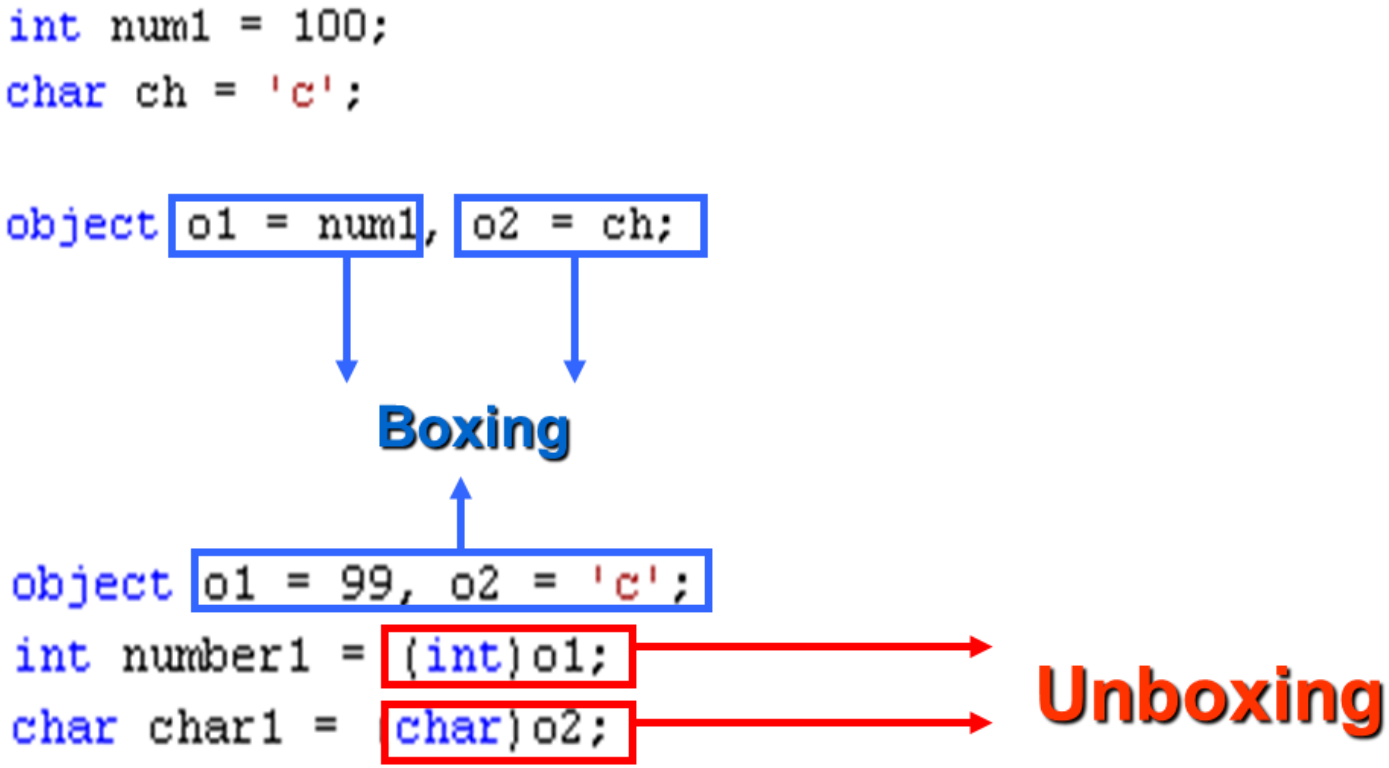

```
int a = 930;
int b = 10;
Console.WriteLine("{0,4}\n+{1,3}\n----\n{2,4}", a, b, a + b);
```

\n: ký tự xuống dòng



Boxing & Unboxing

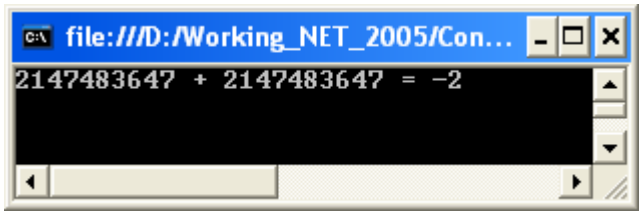
- Kiểu giá trị có thể được chuyển thành kiểu đối tượng
- Boxing
- Unboxing



checked & unchecked

```
int x = int.MaxValue;
int y = int.MaxValue;
int z = unchecked(x + y);

Console.WriteLine("{0} + {1} = {2}",
        x.ToString(),y.ToString(),z.ToString());
```




```
int x = int.MaxValue;
int y = int.MaxValue;
int z = checked(x + y);

Console.WriteLine("{0} + {1} = {2}",
        x.ToString(),y.ToString(),z.ToString());
```

throws OverFlowException

ref, out, params

- ref: tương tự như truyền tham chiếu trong C/C++
- Từ khoá ref phải được dùng lúc gọi hàm
- Các tham số truyền dạng ref phải được khởi tạo giá trị trước

```
public static int Main()
{
    int num1 = 5, num2 = 2;
    Swap(ref num1,ref num2);

    return 0;
}
public static void Swap(ref int a, ref int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

sử dụng ref cho tham số khi gọi hàm

Khai báo ref trước kiểu dữ liệu

ref, out, params

- out: tương tự như ref
- Khác ref là out ko cần khởi tạo giá trị trước khi truyền

```
public static int Main()
{
    int a = 10, b;
    Subtraction(a, out b);

    return 0;
}
public static void Subtraction(int x, out int y)
{
    y = x - 2;
}
```

Dùng trước tham số khi gọi hàm

Khai báo cho tham số

DUYTAN edu.vn

ref, out, params

```
public static void Sum( out int result, params int[] myArray)
{
      result = 0;
     for (int i = 0; i < myArray.Length; i++)
         result += myArray[i];

}
```

Luôn khai báo ở cuối danh sách tham số

```
public static int Main()
{
     int s;
     Sum(out s, 1, 2, 3);
     return 0;
}
```

3 phần tử

```
public static int Main()
{
     int s;
     int[] array = { 11, 22, 33, 44 };
     Sum(out s, array);
     return 0;
}
```

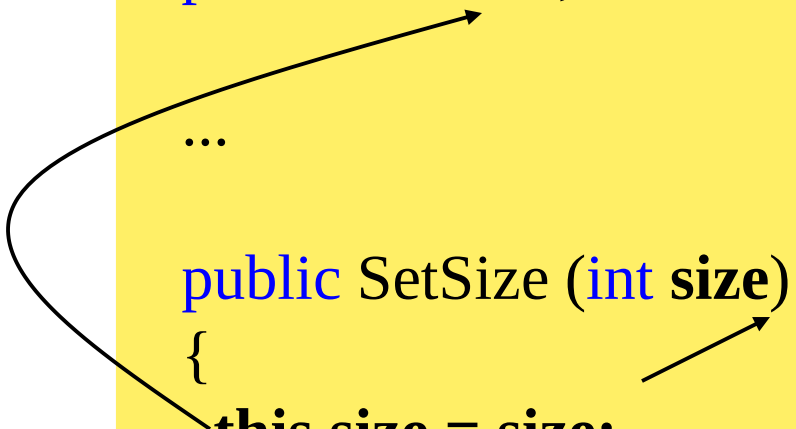
Mảng array

Keyword this

```
public class list
{
   private int size;

   ...

   public SetSize (int size)
   {
     this.size = size;
   }
```



LỆNH

1. Lệnh lặp for, while, do while, foreach
2. Lệnh phân nhánh switch, lệnh nhảy

Video

https://www.youtube.com/watch?time_continue=52&v=mKLRETK9slQ&feature=emb_logo

- Tương tự như C/C++: while, do while, for

```
while <điều kiện>
{
    // phần thân while
}
```

```
do
{
    // phần thân do while
} while <điều kiện>;
```

```
for( khởi tạo biến lặp; <điều kiện theo biến lặp>; thay đổi biến lặp)
{
    // phần thân for
}
```

Phải là giá trị bool: true, false

LỆNH

```
index = 10;
while (index != 0){
   Console.WriteLine(index);
   index--;
}
```

Giá trị {true, false}

```
index = 0;
do{
   Console.WriteLine("Happens at least once");
}while (index < 0);
```

```
for(index = 0; index < 100; index++){
   Console.Write(index);
   Console.Write("\t");
}
```

foreach

Cú pháp

```
foreach( typedata identifier in objectArray)
{
    // thân foreach
}
```

foreach

```
public static int Main()
{
    Int32[] myArray = new Int32[] { 10, 20, 30, 40 };
    foreach (Int32 i in myArray)
    {
        Console.WriteLine(i);
    }
    return 0;
}
```

Chỉ sử dụng biến i cho mỗi lần lặp

=

```
public static int Main()
{
    Int32[] myArray = new Int32[] { 10, 20, 30, 40 };
    for (int i = 0; i < myArray.Length; i++)
    {
        Console.WriteLine(myArray[i]);
    }
    return 0;
}
```

Sử dụng chỉ số mảng như bình thường

DUYTAN edu.vn

switch

```
switch (country)
{
    case "Germany":
    case "Austria":
    case "Switzerland":
        language = "German";
        break;
    case "England":
    case "USA":
        language = "English"; break;
    case null:
        Console.WriteLine("no country specified");
        break;
    default:
        Console.WriteLine("don't know language of {0}", country);
        break;
}
```

switch

- Biểu thức switch gồm: kiểu số, ký tự, enum và chuỗi
- Sử dụng break, goto, return để điều khiển luồng thực thi
- Nếu ko nhãn nào phù hợp → default
- Nếu ko có default → thực hiện lệnh sau switch

Jump

- break
  - Thoát khỏi vòng lặp
- continue
  - Qua bước lặp kế
- goto
  - Nhảy đến nhãn
  - Sử dụng goto case <**expression**>, trong switch

```
for(int i=0; i < myArr.Length;i++)
{
    if (myArr[i] <= -1)
        break;
    if (myArr[i] == 0)
        continue;
    else
        myArr[i]++;
}
```

return

- Thoát khỏi hàm void

```
void Func1(int x)
{
    if (x == 0)
        return;
    ...
}
```

- Trả về 1 giá trị của hàm

```
int max(int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

Câu hỏi

1. Sự khác nhau giữa kiểu dữ liệu giá trị và kiểu dữ liệu tham chiếu? Kiểu chuỗi, kiểu lớp trong C# là kiểu dữ liệu nào?
2. So sánh cấu trúc điều kiện if và switch
3. So sánh cấu trúc lặp while và do… while
4. Cho biết các lệnh rẻ nhánh có điều kiện và lệnh rẻ nhánh không điều kiện

Bài tập

Bài 1: Viết chương trình tính

S= 1 +2 + 3 +… +n với n nhập từ bàn phím

Bài 2:viết chương trình tính tổng bình phương của các số sau

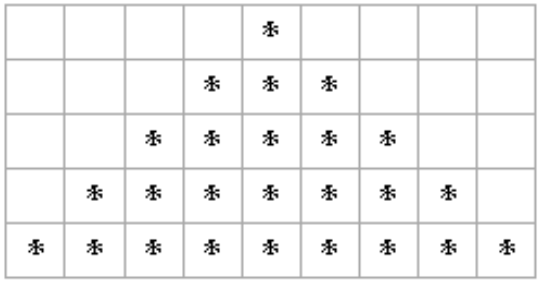
S= $1^2 + 2^2 + 3^2 + \ldots + n^2$ (n nhập từ bàn phím)

Bài 3: Viết chương trình tính giai thừa của n (với n nhập từ bàn phím)

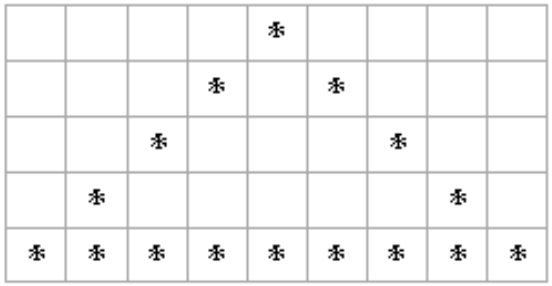
Bài tập

Bài 4: Viết chương trình xuất ra màn hình tam giác cân hợp bởi các ký tự '*' với chiều cao =5(hình 1

Hình 1



Hình 2



Bài 5: Viết chương trình xuất ra màn hình tam giác cân (rỗng) hợp bởi các ký tự '*' với chiều cao nhập từ bàn phím (hình 2)

Câu hỏi

1. Biến kiểu Boolean được tự động khởi tạo với giá trị?
2. Liệt kê các kiểu giá trị trong C#?
3. Điểm khác nhau giữa lệnh break và lệnh continue?
4. Theo quyền ưu tiên của các toán tử trong C#, toán tử nào có quyền ưu tiên cao nhất?
5. Dot Operator (Toán tử .) là gì?

YOUTUBE

1. https://www.youtube.com/watch?v=wfWxdh-_k_4
2. https://www.youtube.com/watch?v=gfkTfcpWqAY
3. https://www.youtube.com/watch?v=GhQdlIFylQ8
4. https://www.youtube.com/watch?v=SXmVym6L8dw&list=PLAC325451207E3105