LẬP TRÌNH HƯỚNG SỰ KIỆN

Giảng viên: ThS. Phan Thanh Toàn

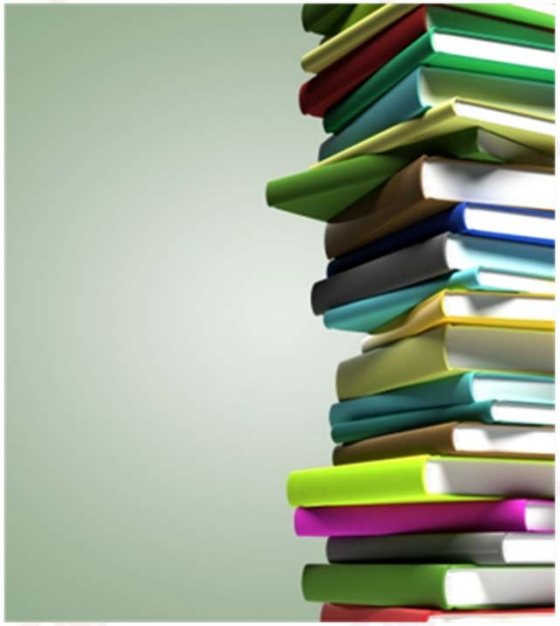
BÀI 1
TỔNG QUAN VỀ .NET FRAMEWORK VÀ NGÔN NGỮ LẬP TRÌNH C#

Giảng viên: ThS. Phan Thanh Toàn

MỤC TIÊU BÀI HỌC

- Liệt kê được các đặc trưng cơ bản của .Net FrameWork.
- Phân biệt được các khái niệm hằng, biến, biểu thức.
- Viết được cú pháp các lệnh cơ bản trong C# để khai báo biến, hằng…
- Vận dụng được các cấu trúc lập trình cơ bản của C# để giải quyết một số bài toán đơn giản.
- Phân biệt được các kiểu dữ liệu như mảng, xâu kí tự, kiểu liệt kê.

CÁC KIẾN THỨC CẦN CÓ



Để học được môn học này, sinh viên phải học xong các môn học:

- Cơ sở lập trình;
- Lập trình hướng đối tượng;
- Cơ sở dữ liệu;
- Hệ quản trị cơ sở dữ liệu SQL Server.

HƯỚNG DẪN HỌC

- Đọc tài liệu tham khảo.
- Thảo luận với giáo viên và các sinh viên khác về những vấn đề chưa hiểu rõ.
- Trả lời các câu hỏi của bài học.

CẤU TRÚC NỘI DUNG

1.1. GIỚI THIỆU TỔNG QUAN VỀ .NET FRAMEWORK VÀ MỘT SỐ KHÁI NIỆM CƠ BẢN

1.1.1. Tổng quan về .Net FrameWork

1.1.2. Kiểu dữ liệu

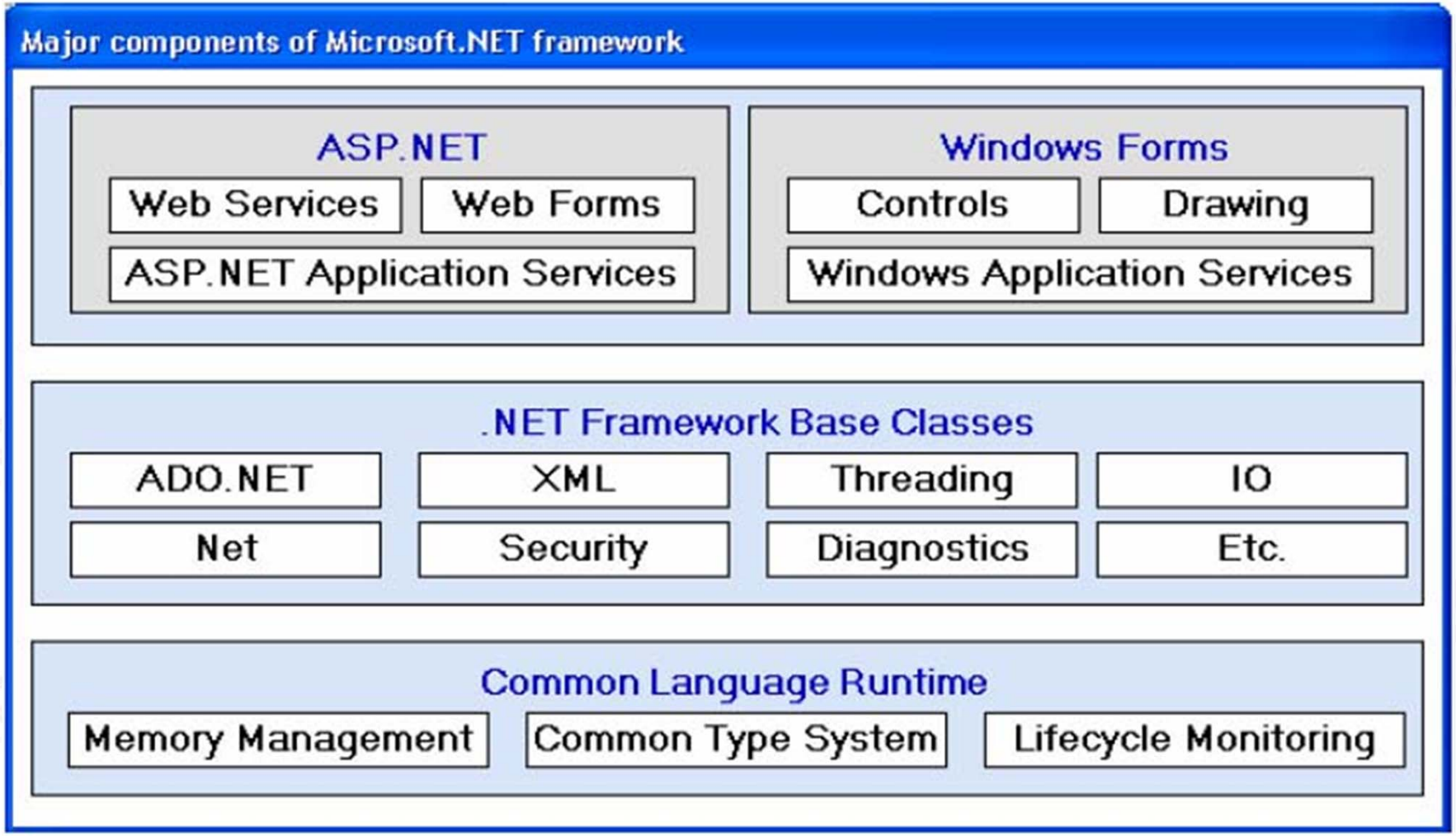
1.1.3. Biến và hằng

1.1.1. TỔNG QUAN VỀ .NET FRAMEWORK

- .NET được phát triển từ năm 1998 bởi công ty Microsoft. Mục tiêu là tạo ra một hệ thống hỗ trợ phát triển các ứng dụng trên nền tảng công nghệ internet và các ứng dụng phân tán.
- Microsoft .NET hỗ trợ tạo ra các sản phẩm có thể chạy trên nhiều nền tảng công nghệ và độc lập với phần cứng.
- .NET FrameWork cung cấp khoảng 5000 lớp đối tượng hỗ trợ các dịch vụ từ hệ điều hành.
- .NET FrameWork cung cấp 2 thành phần chính: Các lớp cơ sở (.NET FrameWork base class) và sử dụng ngôn ngữ chung (Common Language Runtime).
- .NET FrameWork cung cấp tập các hàm API giúp các lập trình viên thuận tiện trong khai thác và sử dụng.

1.1.1. TỔNG QUAN VỀ .NET FRAMEWORK (tiếp theo)

- Thành phần chính của .NET FrameWork:



1.1.1. TỔNG QUAN VỀ .NET FRAMEWORK (tiếp theo)

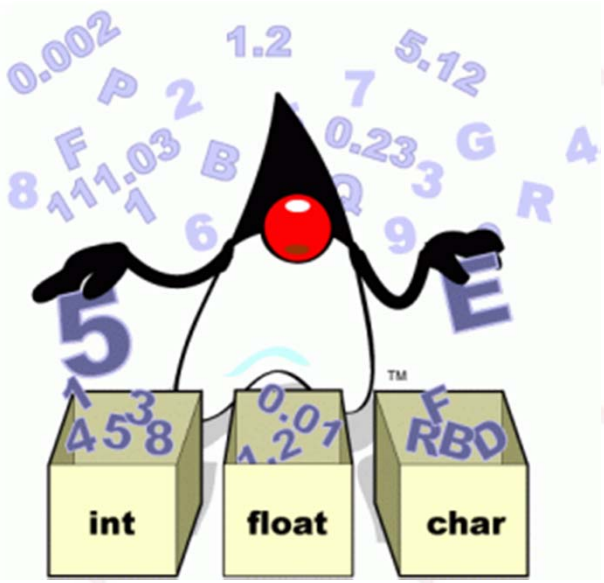
- .NET Application có 2 loại:
  - ASP.NET gồm web form và web service;
  - Window form.
- Common Language Runtime (CLR): là thành phần kết nối các phần khác nhau trong .NET FrameWork với hệ điều hành, CLR có vai trò quản lí việc thực thi các ứng dụng viết bằng .NET trên môi trường window.
- Bộ thư viện và các lớp đối tượng: .NET FrameWork là một tập hợp các thư viện hỗ trợ lập trình viên, với hơn 5000 lớp đối tượng.
- ADO.NET và XML: Bộ thư viện hỗ trợ thao tác với cơ sở dữ liệu (CSDL) và các dữ liệu phi cấu trúc XML.
- ASP.NET: Bộ công cụ hỗ trợ phát triển các ứng dụng web form.
- Webservice: Bộ công cụ và các dịch vụ internet.
- Window form: Bộ công cụ hỗ trợ phát triển các ứng dụng window form.

1.1.2. KIỂU DỮ LIỆU

- Định nghĩa:

  Kiểu dữ liệu T là một 2-bộ (D, O), trong đó:

  - D: là miền trị, là tập các giá trị của kiểu dữ liệu T;
  - O: là các phép toán trên kiểu dữ liệu T.
- Trong ngôn ngữ lập trình kiểu dữ liệu là tập các giá trị mà một biến thuộc kiểu đó có thể nhận và các thao tác xử lí trên kiểu đó.
- Kiểu dữ liệu trong .NET được mô tả chi tiết trong một cấu trúc gọi là Common Type System (CTS).



1.1.2. KIỂU DỮ LIỆU (tiếp theo)

- Các kiểu dữ liệu trong .NET được chia thành nhiều loại:
  - Kiểu giá trị;
  - Kiểu tham chiếu;
  - Kiểu do người dùng định nghĩa;
  - Kiểu liệt kê.
- Mỗi kiểu dữ liệu trong .NET thực chất là một đối tượng với các thuộc tính và phương thức riêng.
- Các kiểu dữ liệu giá trị được xây dựng sẵn bởi .NET gồm có một số kiểu cơ bản cụ thể như sau:

1.1.2. KIỂU DỮ LIỆU (tiếp theo)

Kiểu/ Alias C#	Kích thước	Miền giá trị (Range)
System.SByte/sbyte	1 byte	-128 → 127
System.Byte/byte	1 byte	0 → 255
System.Short/short	2 byte	-32768 → 32767
System.Integer/int	4 bytes	-2147483648 → 2147483647
System.UInteger/uint	4 bytes	0 → 4294967295
System.Long/long	8 bytes	-9223372036854775808 → 9223372036854775807
System.Single/float	4 bytes	-3.402823E+38 → 3.402823E+38
System.Double/double	8 bytes	-1.7976931348623E+308 → 1.7976931348623E+308
System.Decimal/decimal	16 bytes	-79228162514264337593543950335 → 79228162514264337593543950335
System.Char/char	2 bytes	N/A
System.Boolean/bool	4 bytes	N/A
System.DateTime (Date/date)	8 bytes	1/1/0001 12:00:00 AM → 12/31/9999 11:59:59 PM

1.1.2. KIỂU DỮ LIỆU (tiếp theo)

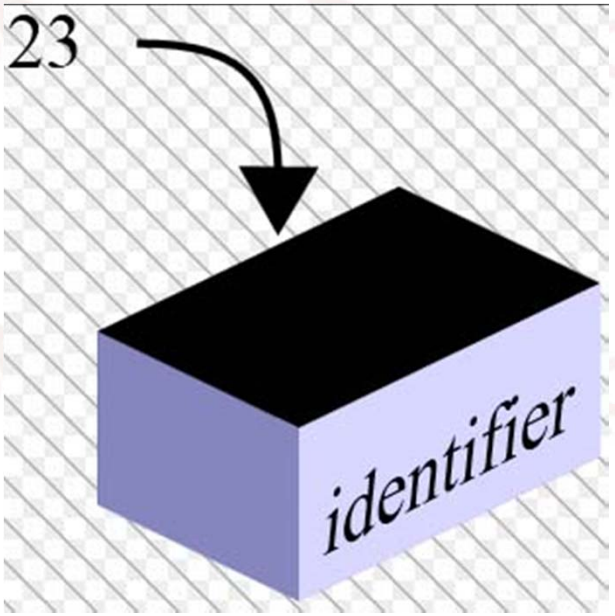
Các kiểu dữ liệu tham chiếu trong .NET

Kiểu	Miền giá trị (Range)
System.Object	Kiểu tổng quát
System.String	Dữ liệu dạng text
System.Text.StringBuilder	Dữ liệu dạng Dynamic text
System.Array	Mảng dữ liệu
System.IO.Stream	Bộ đệm (Buffer) cho tập tin, thiết bị
System.Exception	Kiểm soát hệ thống và trình ứng dụng

1.1.3. BIẾN VÀ HẰNG

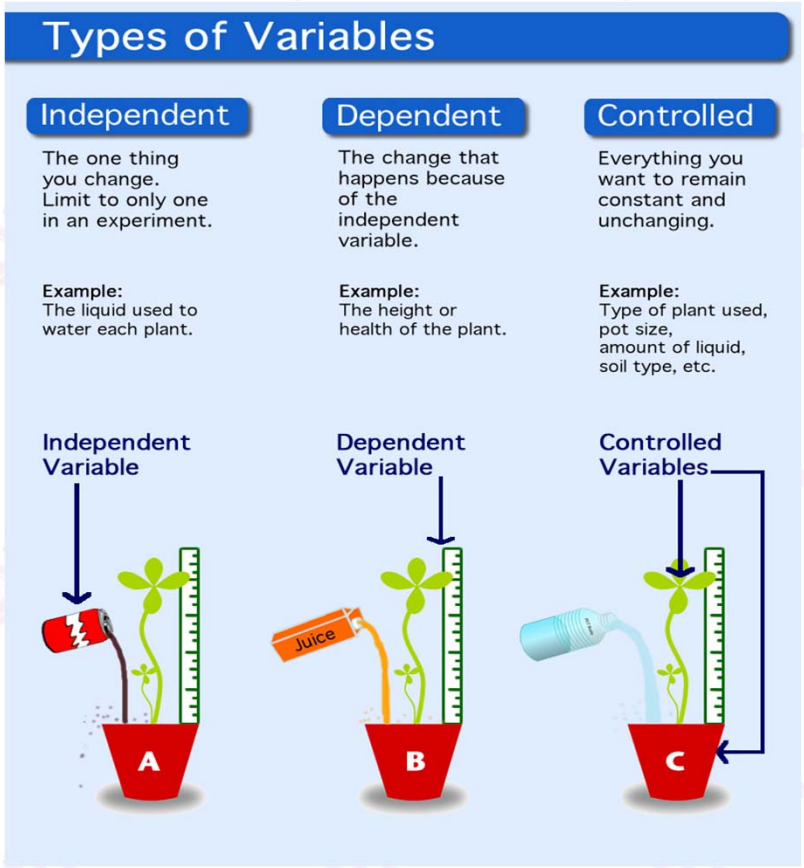
a. Biến

- Khái niệm: Biến là một đại lượng có giá trị thay đổi được khi thực hiện chương trình. Thực chất biến là một ô nhớ có khả năng lưu trữ các giá trị khác nhau.
- Mỗi biến luôn có một kiểu dữ liệu xác định.
- Các biến có thể lưu trữ các giá trị dạng chuỗi, số, dạng thời gian... tùy thuộc vào kiểu dữ liệu của biến.
- Mỗi biến có một tên xác định (identifier).



1.1.3. BIẾN VÀ HẰNG (tiếp theo)

- Trong kĩ thuật lập trình các biến thường chia thành 3 loại: biến độc lập, biến phụ thuộc và biến kiểm soát.
  - ➢ Biến độc lập (independent variable): là biến được sử dụng để mô tả, đo lường các yếu tố được coi là nguyên nhân hay có ảnh hưởng đến đối tượng.
  - ➢ Biến phụ thuộc (dependent variable): là hậu quả của sự tác động các biến độc lập, là biến được sử dụng để mô tả, đo lường các đối tượng.
  - ➢ Biến kiểm soát (controlled variabe): là các biến sử dụng để kiểm soát các thay đổi đến đối tượng.



1.1.3. BIẾN VÀ HẰNG (tiếp theo)

- Cách khai báo biến trong C#:

  Cú pháp: <Kiểu dữ liệu > <Tên biến>;

- Ví dụ:

  int x, y;

  float a, b;

  char ch;

- Các qui tắc đặt tên biến:
  - Tên biến không chứa dấu cách (khoảng trống);
  - Tên biến không trùng với từ khóa;
  - Tên biến không chứa các kí tự đặc biệt: #, $…
  - Tên biến không bắt đầu bằng số;
  - Tên biến phân biệt chữ hoa, chữ thường.

1.1.3. BIẾN VÀ HẰNG (tiếp theo)

- Đặc điểm:
  - Khi khai báo một biến trong C#, biến được xem như một đối tượng.
  - Mỗi đối tượng sẽ có 2 thành phần cơ bản là: thuộc tính (property) và phương thức (method).
  - Sau khi khai báo biến đối tượng ta có thể truy xuất đến các thuộc tính và phương thức theo cú pháp sau:

    <Tên biến>.<Tên thuộc tính>

    <Tên biến>.<Tên phương thức>

    Ví dụ:

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

1.1.3. BIẾN VÀ HẰNG (tiếp theo)

b. Hằng

- Khái niệm: là đại lượng có giá trị không thay đổi khi thực hiện chương trình.
- Cú pháp khai báo hằng:

  const <Kiểu dữ liệu> <Tên hằng> = giá trị;
- Ví dụ:

  const double PI = 3.14;
- Biểu thức: là sự kết hợp của các toán tử và các toán hạng. Mỗi biểu thức luôn trả về một giá trị duy nhất.

  Ví dụ: x*5 + 2;

1.1.3. BIẾN VÀ HẰNG (tiếp theo)

- Các toán tử trong C#:
  - Toán tử số học:

Toán tử	Mô tả
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư

1.1.3. BIẾN VÀ HẰNG (tiếp theo)

- Toán tử gán:

Ký hiệu	Mô tả	Mô tả
=	Gán toán hạng thứ hai cho toán hạng thứ nhất.	
+=	Cộng hoặc nối chuỗi toán hạng sau vào toán hạng đầu và gán kết quả cho toán hạng đầu.	Tăng biến a lên 1 a += 1 ⇔ a = a+1
-=	Trừ toán hạng sau khỏi toán hạng đầu và gán hiệu cho toán hạng đầu.	Giảm biến a đi 1 a -= 1 ⇔ a = a-1
*=	Nhân hai toán hạng với nhau và gán tích cho toán hạng đầu.	Gấp đôi biến a a *= 2 ⇔ a = a*a
/=	Chia toán hạng đầu cho toán hạng sau và gán thương cho toán hạng đầu.	Chia đôi biến a a /= 2 ⇔ a = a/2

1.1.3. BIẾN VÀ HẰNG (tiếp theo)

- Toán tử luận lí (logic):

Toán tử	Mô tả
!	Phép toán phủ định trên một giá trị luận lý
&&	Phép toán luận lý VÀ (AND) trên hai giá trị. Kết quả trả về là true nếu cả hai giá trị luận lý đều là true.
||	Phép luận lý HOẶC (OR) trên hai giá trị. Kết quả trả về là false nếu cả hai giá trị luận lý đều là false.

- Các toán tử với bit:

Tên	Toán tử	Mô tả
Logical NOT	~a	Phép đảo giá trị bit.
Logical AND	a & b	Trả về 1 nếu cả hai bit đều là 1, ngược lại trả về 0.
Logical OR	a | b	Trả về 0 nếu cả hai bit đều là 0, ngược lại trả về 1.
Logical XOR	a ^ b	Trả về 1 nếu và chỉ nếu một trong hai bit là 1, ngược lại trả về 0.

1.1.3. BIẾN VÀ HẰNG (tiếp theo)

- Các toán tử so sánh:

Toán tử	Mô tả
==	Bằng
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
>	Lớn hơn
<	Nhỏ hơn
!=	Khác
<	Nhỏ hơn

1.2. CẤU TRÚC LẬP TRÌNH CƠ BẢN TRONG C#

1.2.1. Cấu trúc rẽ nhánh

1.2.2. Cấu trúc lặp

1.2.3. Xử lí lỗi trong C#

1.2.1. CẤU TRÚC RẼ NHÁNH

- Một chương trình là một dãy các câu lệnh được viết theo trình từ, hầu hết các ngôn ngữ lập trình sẽ thực hiện các lệnh trong chương trình một cách tuần tự từ trên xuống, khi cần điều khiển chương trình thực hiện theo các điều kiện khác nhau ta sẽ sử dụng cấu trúc rẽ nhánh trong chương trình.
- Cấu trúc rẽ nhánh là cấu trúc mà có 1/1 khối lệnh sẽ được thực hiện hoặc bỏ qua tùy theo giá trị đúng/sai của một biểu thức điều kiện.
- Cú pháp lệnh rẽ nhánh:

  Trong C# lệnh rẽ nhánh được thể hiện qua lệnh if với cú pháp như sau:

  - Dạng 1:
    - Cú pháp:

```
if    (đk)
        S;
```

Trong đó:

if: là từ khóa;

đk: là biểu thức điều kiện trả về giá trị true hoặc false;

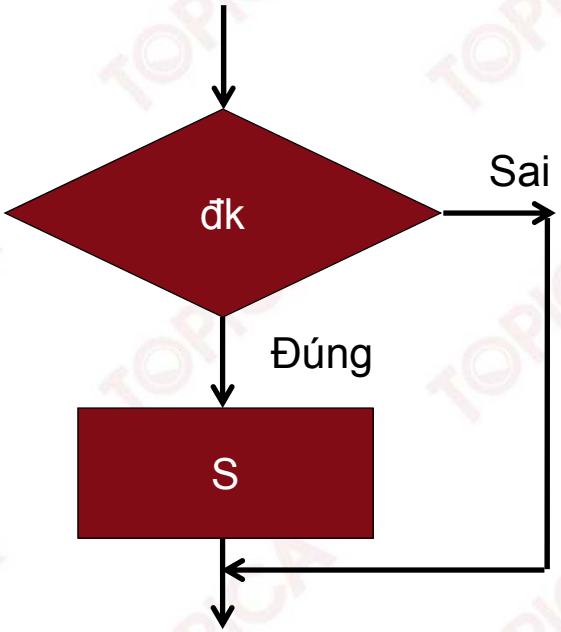
S: là lệnh đơn hoặc lệnh ghép.

1.2.1. CẤU TRÚC RẼ NHÁNH (tiếp theo)

- Hoạt động lệnh if (như hình vẽ)

  Ví dụ: Chương trình kiểm tra số nguyên có phải số chẵn không?

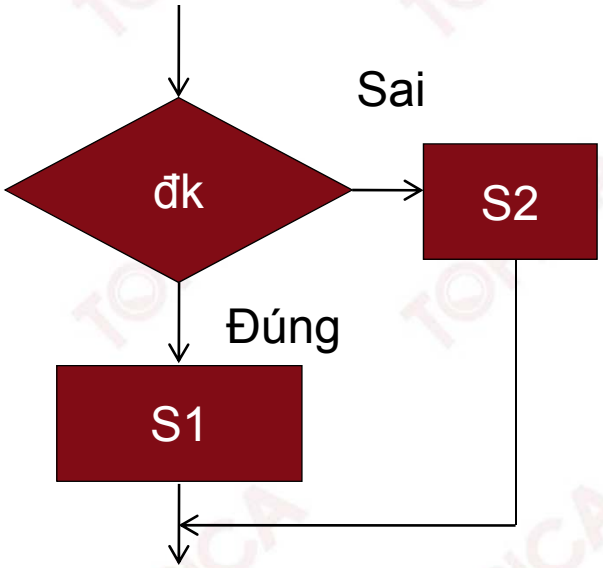
```
int x;
    Console.Write("NHAP SO NGUYEN:");
    x= int.Parse(Console.ReadLine());
    if (x % 2 == 0)
    Console.WriteLine("{0} LA SO CHAN", x);
```



1.2.1. CẤU TRÚC RẼ NHÁNH (tiếp theo)

- Dạng 2:
  - Cú pháp:

```
if (đk)
  S1;
else
  S2;
```



  - Hoạt động như hình vẽ

Ví dụ: Kiểm tra tính chẵn lẻ của số nguyên

```
int x;
Console.Write("NHAP SO NGUYEN:");
x= int.Parse(Console.ReadLine());
if (x % 2 == 0)
Console.WriteLine("{0} LA SO CHAN", x);
else
Console.WriteLine("{0} LA SO LE", x);
```

1.2.1. CẤU TRÚC RẼ NHÁNH (tiếp theo)

- Chú ý:
  - Có thể sử dụng lồng nhau nhiều lệnh if để giải quyết các bài toán với điều kiện phức tạp.

    Dạng lệnh:

```
if (đk 1)
        S1;
else if(đk 2)
        S2;
else if (đk 3)
        S3;
.....................
```

  - Có thể sử dụng các toán tử logic kết hợp với lệnh if để giải quyết các bài toán với điều kiện phức tạp.
- Ví dụ:

```
if (A > 0 && A < 10)
```

1.2.1. CẤU TRÚC RẼ NHÁNH (tiếp theo)

- Lệnh switch: khi bài toán có nhiều nhánh lựa chọn rời rạc ta có thể sử dụng cấu trúc switch theo cú pháp sau:
  - Cú pháp:

```
switch (biểu thức chọn)
{
        case  <giá trị 1>:
                S1;
                break;
        case  <giá trị 2>:
                S2;
                break;
        ……..
        default:
                Sn+1 ;
        break;
}
```

1.2.1. CẤU TRÚC RẼ NHÁNH (tiếp theo)

Trong đó:

- switch, default, case: là các từ khóa;
- biểu thức chọn: là một biểu thức có giá trị rời rạc;
- S1, S2,…., S_{n+1} là các lệnh đơn/khối lệnh.

➢ Hoạt động:

- Tính giá trị của biểu thức chọn.
- Nếu biểu thức chọn có giá trị trùng với giá trị sẽ thực hiện lệnh S_i tương ứng, nếu có lệnh break sẽ kết thúc lệnh switch, nếu không có lệnh switch. sẽ tiếp tục thực hiện các nhánh lệnh tiếp theo cho đến khi gặp lệnh break hoặc dấu } kết thúc lệnh switch.
- Nếu giá trị của biểu thức chọn không trùng với bất kì giá trị nào trong các nhánh case sẽ thực hiện lệnh sau default và kết thúc lệnh switch.

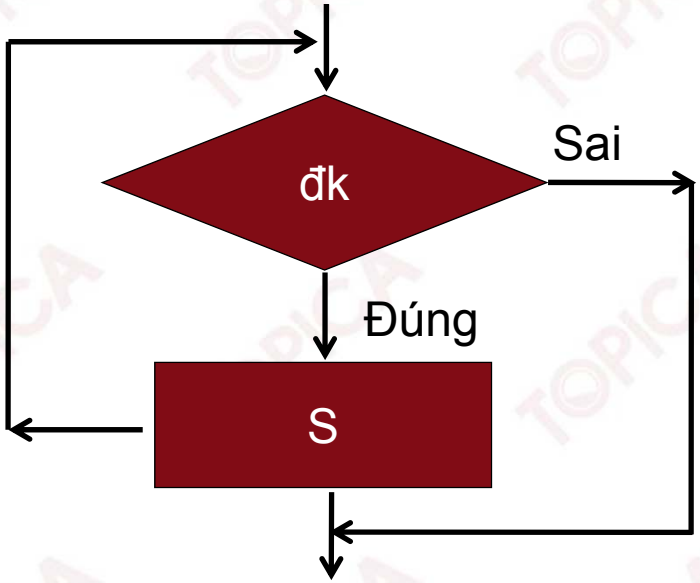
1.2.2. CẤU TRÚC LẶP

- Cấu trúc lặp là cấu trúc trong đó có 1 hoặc 1 nhóm lệnh được thực hiện lặp đi lặp lại nhiều lần.
- Trong C# có một số lệnh lặp: for, while, do while
  - Lệnh while
    - Cú pháp:

```
while (đk)
        S;
```

Trong đó:
while là từ khóa;
Đk là biểu thức điều kiện;
S là khối lệnh.

- Hoạt động hình 3.



1.2.2. CẤU TRÚC LẶP

Ví dụ: Chương trình tính tổng 1+2+…+N, với N nhập từ bàn phím

```
int n, i;
nt tong = 0;
Console.Write("NHAP SO NGUYEN N=");
n = int.Parse(Console.ReadLine());
i = 1;
while (i <= n)
{
   tong += i;
   i++;
}
Console.WriteLine("TONG LA: " + tong);
```

Chú ý: có thể sử dụng lệnh break để kết thúc vòng lặp một cách vô điều kiện.

1.2.2. CẤU TRÚC LẶP (tiếp theo)

- Lệnh do while
  - Cú pháp:

```
do {
S;
} while (đk);
```

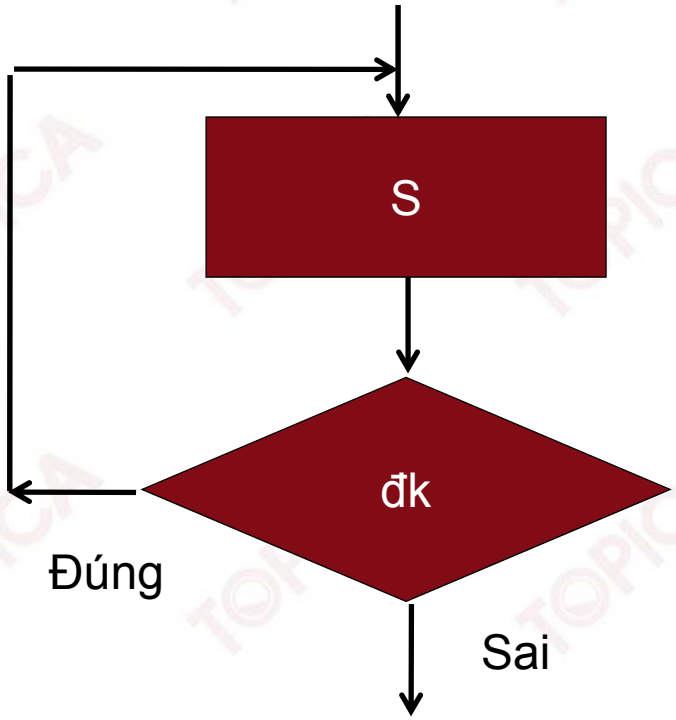
Trong đó:

do, while là các từ khóa;

đk là biểu thức điều kiện;

S là khối lệnh.

  - Hoạt động như hình vẽ.



1.2.2. CẤU TRÚC LẶP (tiếp theo)

Ví dụ: Chương trình tính tổng 1+2+..+N, N nhập từ bàn phím

```
int n, i;
int tong = 0;
Console.Write("NHAP SO NGUYEN N=");
n = int.Parse(Console.ReadLine());
i = 1;
do
{
    tong += i;
    i++;
} while (i <= n);
Console.WriteLine("TONG LA: " + tong);
```

1.2.2. CẤU TRÚC LẶP (tiếp theo)

- Lệnh for
  - Cú pháp:

```
for( bt1; bt2; bt 3)
        S;
```

Trong đó:

for: là từ khóa;

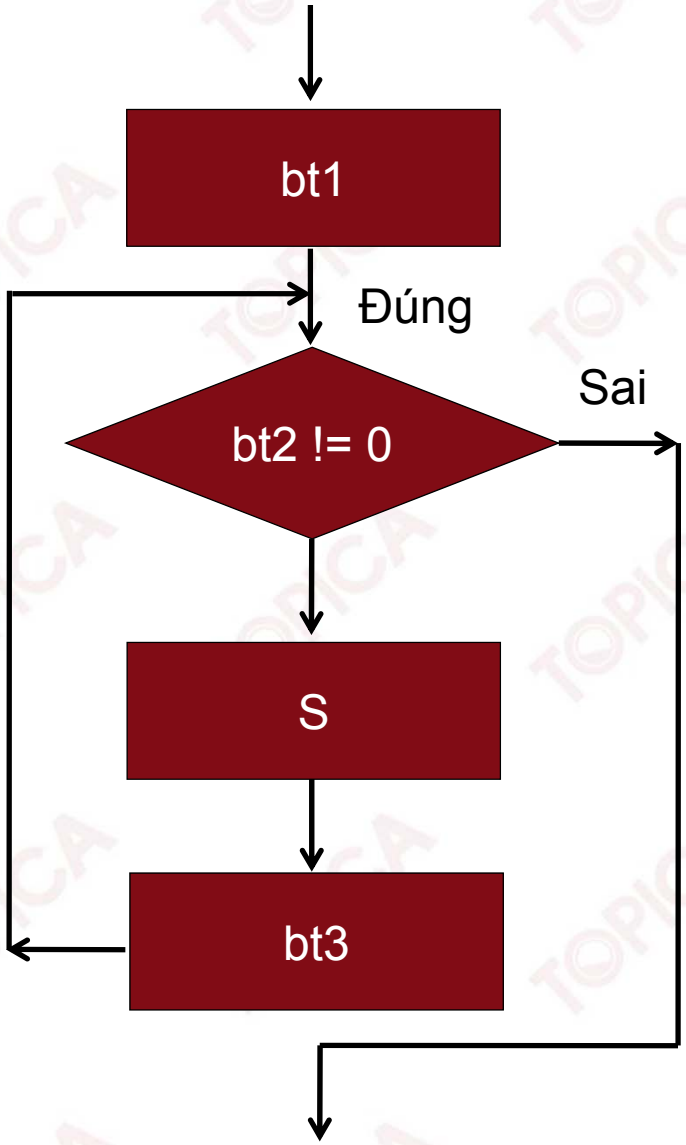
bt1, bt2, bt3: là các biểu thức;

S: là khối lệnh.

  - Hoạt động như hình vẽ.
  - Nhận xét:

Biểu thức bt1 chỉ được thực hiện duy nhất 1 lần (biểu thức khởi tạo).

Biểu thức bt2, bt3 được lặp đi lặp lại nhiều lần, bt2 là biểu thức điều kiện quyết định vòng lặp được lặp tiếp hay kết thúc.



1.2.2. CẤU TRÚC LẶP (tiếp theo)

Ví dụ: Chương trình tính tổng 1+2+..+N, N nhập từ bàn phím

```
int n, i;
 int tong = 0;
 Console.Write("NHAP SO NGUYEN N=");
 n = int.Parse(Console.ReadLine());
 for (i = 1; i <= n; i++)
     tong += i;
 Console.WriteLine("TONG LA: " + tong);
```

Chú ý: có thể sử dụng cấu trúc foreach để duyệt các phần tử của một tập hợp.

Cú pháp:

```
foreach ( <kiểu dữ liệu> <phần tử> in <tập hợp> )
{
    Tập lệnh;
}
```

1.2.3. XỬ LÝ LỖI TRONG C#

- Phân loại lỗi:

  Khi xây dựng chương trình thường gặp một số loại lỗi sau:

  - Lỗi cú pháp (Syntax error):
    - Xuất hiện khi viết code chương trình;
    - Được thông báo khi dịch chương trình;
    - Nguyên nhân: do viết các lệnh sai cú pháp.
  - Lỗi thực thi (Runtime error):
    - Xảy ra khi chạy chương trình;
    - Khó xác định hơn lỗi cú pháp;
    - Nguyên nhân: Không xử lí hết các tình huống khi lập trình, lỗi chia cho 0, lỗi không mở được kết nối tới CSDL, mở một tệp tin không tồn tại…
  - Lỗi logic:
    - Xảy ra khi thực hiện chương trình;
    - Thể hiện dưới dạng thu được các kết quả không mong đợi;
    - Nguyên nhân: Sai trong thuật giải và các quá trình xử lý dữ liệu.

1.2.3. XỬ LÝ LỖI TRONG C#

- Các phương pháp xử lí lỗi:
  - Dò và sửa lỗi thủ công: Khi có lỗi phát sinh ta tự tìm dòng lệnh bị lỗi và sửa cho đúng cú pháp, học thực hiện các lệnh một cách tuần tự để tìm lỗi và sửa.
  - Sử dụng lệnh Throw: Thường được sử dụng để báo hiệu sự cố xảy ra của một tình trạng bất thường (ngoại lệ - exception) trong chương trình thực thi.

    Cú pháp:

    throw new <loại ngoại lệ>(thông báo lỗi)

    Ví dụ:

```
string s = null;
Console.WriteLine("Example of proccessing exception");
if (s == null)
{
    throw new ArgumentNullException();
}
Console.WriteLine("The string is null");
```

1.2.3. XỬ LÝ LỖI TRONG C# (tiếp theo)

- Sử dụng mệnh đề bẫy lỗi try catch

Cú pháp:

```
try {
        Khối lệnh thực thi;
}
catch(Exception)
{
        Khối lệnh xử lí lỗi;
}
finally
{
        Khối lệnh luôn được thực hiện;
}
```

1.2.3. XỬ LÝ LỖI TRONG C# (tiếp theo)

Ví dụ:

```
int x,y=10;
int z = 0;
try
 {
    x = y / z;
    Console.WriteLine("KET QUA PHEP CHIA LA: {0}", x);
 }
catch(Exception e)
  {
   Console.WriteLine("LOI CHIA CHO 0");
  }
```

1.3. KIỂU DỮ LIỆU CÓ CẤU TRÚC TRONG C#

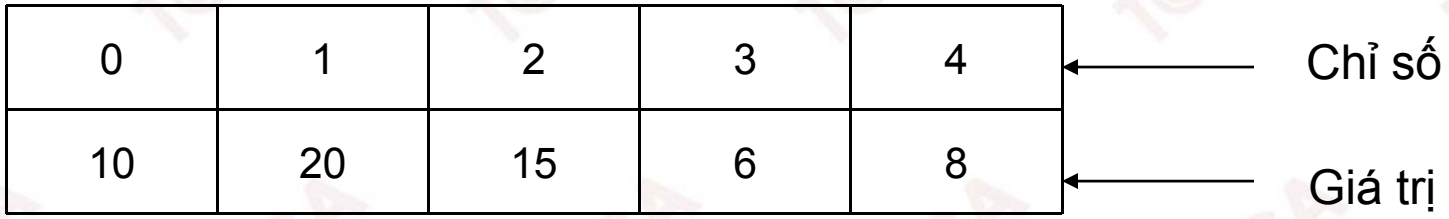
1.3.1. Dữ liệu kiểu mảng

1.3.2. Dữ liệu kiểu xâu kí tự

1.3.3. Dữ liệu kiểu liệt kê

1.3.1. DỮ LIỆU KIỂU MẢNG

- Định nghĩa: Mảng là kiểu dữ liệu bao gồm một dãy liên tiếp, hữu hạn các phần tử có chung nhau một kiểu dữ liệu.
- Mỗi phần tử trong mảng được xác định bởi chỉ số phần tử của mảng.
- Trong C# chỉ số phần tử mảng được bắt đầu từ 0.



Khai báo mảng:

Cú pháp:

<Kiểu dữ liệu> [] <Tên biến mảng>;

Ví dụ:

int[] a;

Khai báo và khởi tạo kích thước cho mảng theo cú pháp sau:

<Kiểu dữ liệu>[] <tên biến> = new <kiểu dữ liệu>[Số phần t];

Ví dụ:

int [] x = new int [10]; //Tạo một mảng số nguyên gồm 10 phần tử.

1.3.1. DỮ LIỆU KIỂU MẢNG

- Truy xuất các phần tử mảng:
- Cú pháp:

  <Tên biến mảng>[chỉ số];
- Chỉ số là một số nguyên từ 0 đến n-1 (với n là số phần tử của mảng)

  Ví dụ:

  int[] x = new int[10];

  x[0] =10;

  x[1] = 20;
- Lấy chiều dài của mảng: có thể sử dụng thuộc tính Length để lấy chiều dài của mảng.

  Ví dụ: Console.WriteLine(x.Length.ToString());

1.3.1. DỮ LIỆU KIỂU MẢNG (tiếp theo)

- Kỹ thuật duyệt mảng không điều kiện:

  for (int i=0 ; i< <tên biến mảng>.Length; i++)

  XuLy(<Tên biến mảng>[i]);

- Kỹ thuật duyệt mảng có điều kiện:

  for (int i=0 ; i< <tên biến mảng>.Length; i++)

  if(đk (<Tên biến mảng>[i]))

  XuLy(<Tên biến mảng>[i]);

  Trong đó: XuLy() và đk() là các hàm

- Ví dụ: viết chương trình nhập một dãy số nguyên và tính tổng các số chẵn, tổng các số lẻ trong dãy số đã nhập.

1.3.1. DỮ LIỆU KIỂU MẢNG (tiếp theo)

```
int[] x=new int[10];
int n;
int tongchan=0;
int tongle=0;
Console.WriteLine("NHAP SO PHAN TU CUA DAY SO:");
n = Convert.ToInt32(Console.ReadLine());
//NHAP DU LIEU CHO CAC PHAN TU CUA MANG
for (int i = 0; i < n; i++)
{
   Console.Write("X[{0}]=",i+1);
   x[i] = Convert.ToInt32(Console.ReadLine());
}
//TINH TONG CAC SO CHAN VA SO LE TRONG MANG
for (int i = 0; i < x.Length; i++)
if (x[i] % 2 == 0) tongchan += x[i];
else tongle += x[i];
Console.WriteLine("TONG CAC SO CHAN LA:{0}",tongchan);
Console.WriteLine("TONG CAC SO LE LA:{0}", tongle);
```

1.3.2. DỮ LIỆU XÂU KÍ TỰ

- Kiểu xâu kí tự là kiểu dữ liệu được xây dựng sẵn trong C#, kiểu xâu bao gồm một dãy liên tiếp các kí tự
- Khai báo biến xâu:

  string <Tên biến>;
- Một số thao tác với xâu:

  + Gán dữ liệu cho xâu:

  <Tên biến> = Xâu kí tự
- So sánh 2 xâu

  String.Compare(string1, string2): so sánh 2 xâu string1 với string2, nếu string1 < string2 kết quả là -1, nếu string1=string2 kết quả là 0, nếu string1 > string2 kết quả là 1.
- Nối xâu kí tự: String.Concat(string1, string2) kết quả trả về một xâu kí tự là ghép string1 với string2
- Sao chép xâu: string <xâu kết quả> = String.Copy(string1);

1.3.2. DỮ LIỆU XÂU KÍ TỰ

- Chèn một xâu vào một xâu kí tự:

  string <xâu kết quả> = <xâu gốc>. Insert(<Vị trí> , “Xâu cần chèn”);
- Xóa bỏ một số kí tự từ xâu kí tự

  string <xâu kết quả> = <xâu gốc>. Remove(<Vị trí>, số kí tự cần xóa);
- Chuyển xâu thành chữ hoa:

  string <xâu kết quả> = <xâu gốc>. ToUpper()
- Chuyển xâu thành chữ thường

  string <xâu kết quả> = <xâu gốc>. ToLower()

1.3.3. DỮ LIỆU KIỂU LIỆT KÊ

- Định nghĩa: kiểu liệt kê (Enumeration) là kiểu dữ liệu có một số hữu hạn các phần tử được giới hạn bên trong một tập các tên tượng trưng
- Khai báo kiểu liệt kê

  enum <Tên kiểu enum>{danh sách các giá trị};
- Ví dụ: chương trình tạo ra kiểu liệt kê Season gồm 4 giá trị và duyệt dữ liệu trong kiểu liệt kê

```
enum Season
{
    Spring, Summer, Fall, Winter
};
//Lấy giá trị trong enum
Season currentSeason = Season.Fall;
Console.WriteLine(currentSeason);
currentSeason++;
Console.WriteLine(currentSeason);
```

TÓM LƯỢC CUỐI BÀI

Trong bài này, chúng ta đã nghiên cứu các nội dung chính sau:

- Tổng quan về công nghệ .NET FRAMEWORK;
- Khái niệm hằng, biến, biểu thức và câu lệnh trong ngôn ngữ C#;
- Các cấu trúc lập trình cơ bản trong C#;
- Một số kiểu dữ liệu có cấu trúc trong C#.