TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN TIN HỌC VÀ KTTT

# LẬP TRÌNH NÂNG CAO

## Các cấu trúc điều khiển

Giảng viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn

# Các cấu trúc điều khiển trong C++

- Cấu trúc tuần tự
- Cấu trúc rẽ nhánh
- Cấu trúc lặp

# Các cấu trúc điều khiển trong C++

- **Cấu trúc tuần tự**: Là cấu trúc mà trong đó các lệnh được thực hiện một cách tuần tự theo thứ tự xuất hiện trong chương trình (trừ khi gặp lệnh rẽ nhánh, hoặc lệnh lặp)

# Các cấu trúc điều khiển trong C++

- **Cấu trúc rẽ nhánh hoặc cấu trúc lặp**: Là cấu trúc mà trong đó có sử dụng 1 lệnh lựa chọn để quyết định việc đổi hướng thực hiện của chương trình

# Các cấu trúc điều khiển trong C++

- Lệnh lựa chọn là các biểu thức điều kiện.
  - Biểu thức điều kiện đơn là biểu thức sử dụng một trong các toán tử quan hệ: >, <, >=, <=, ==, !=, !
  - Biểu thức điều kiện phức là biểu thức gồm từ 2 biểu thức điều kiện đơn trở lên kết hợp với các toán tử logic &&, ||

# Cấu trúc rẽ nhánh

- C++ cung cấp 2 loại câu lệnh lựa chọn:
  - if
  - switch

# Câu lệnh lựa chọn if

- Là câu lệnh cho phép chương trình rẽ sang một trong 2 hướng đúng hoặc sai của biểu thức kiểm tra điều kiện
- Cấu trúc rẽ nhánh if được phát triển ở các dạng:
- if
- if…else

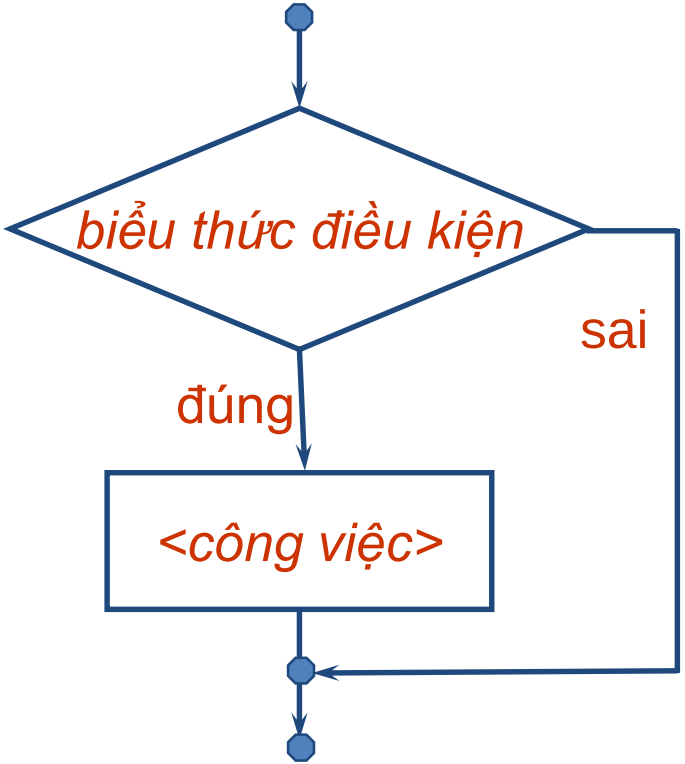
# Cấu trúc rẽ nhánh if dạng 1

- Cú pháp

  **if** (*biểu thức điều kiện*)

  <công việc>

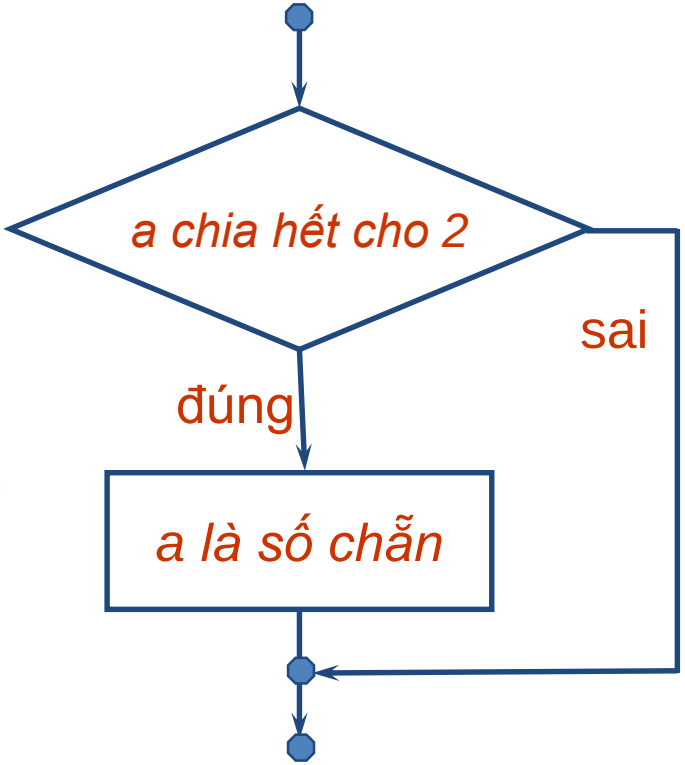
- Ý nghĩa: Nếu *biểu thức điều kiện* có giá trị đúng thì *công việc* được thực hiện, ngược lại không làm gì cả.
- Trong đó: *công việc* là một lệnh đơn hoặc là một khối lệnh nằm trong cặp ngoặc { }




# Cấu trúc rẽ nhánh if dạng 1

- Ví dụ 1: Nếu a chia hết cho 2 thì thông báo a là số chẵn. Ngược lại không thông báo gì hết.

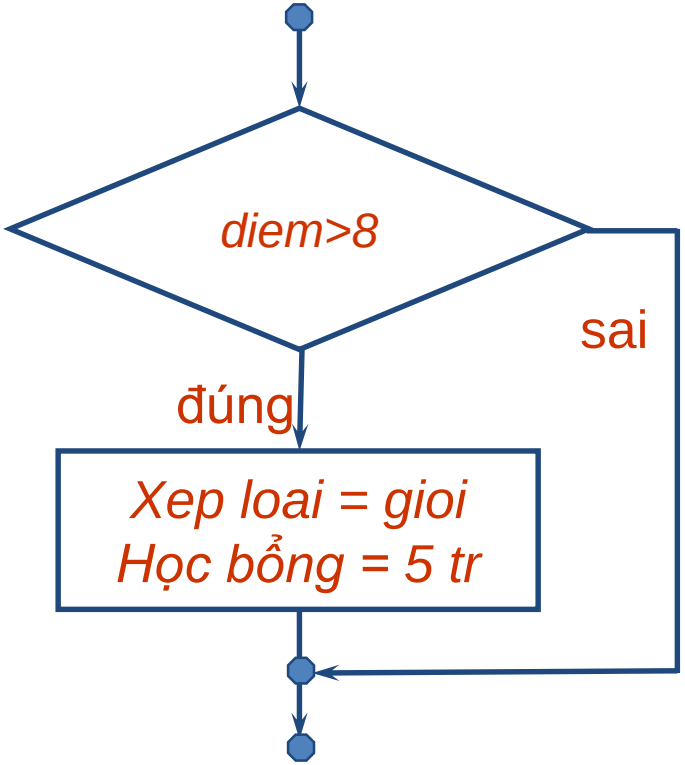
```
if(a%2 == 0)
    cout<<a<<" la so chan";
```




# Cấu trúc rẽ nhánh if dạng 1

- Ví dụ 2: Nếu điểm > 8 thì xếp loại giỏi và thưởng học bổng 5 triệu. Ngược lại không làm gì hết.

```
if(diem>8)
{
    cout<<"xep loai gioi";
    hocbong = 5000000;
}
```




# Cấu trúc rẽ nhánh if dạng 2

- Cú pháp

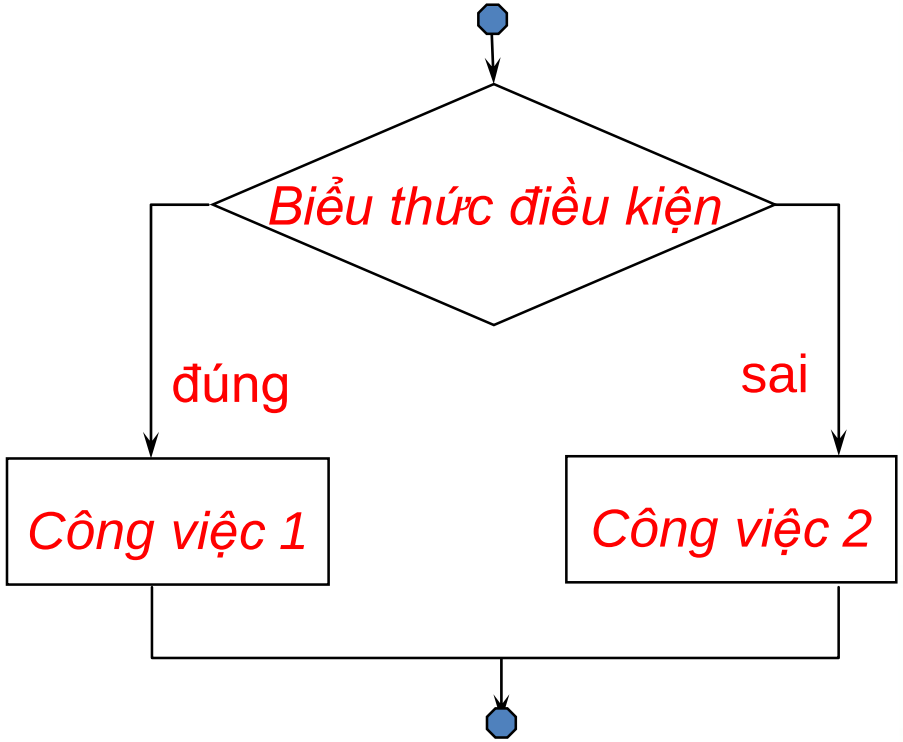
  **if** (*biểu thức điều kiện*)

  *<công việc 1>*

  **else**

  *<công việc 2>*

- Ý nghĩa: Nếu biểu thức có giá trị đúng thì công việc 1 được thực hiện ngược lại thì công việc 2 được thực hiện

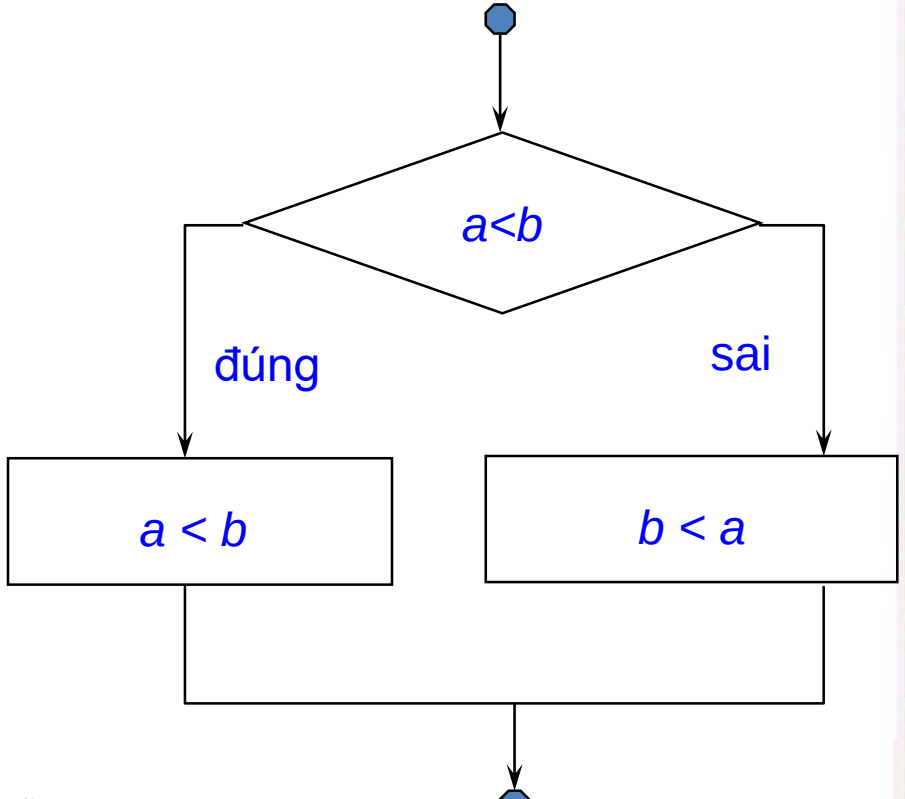



# Cấu trúc rẽ nhánh if dạng 2

- Ví dụ: Cho 2 số a, b.

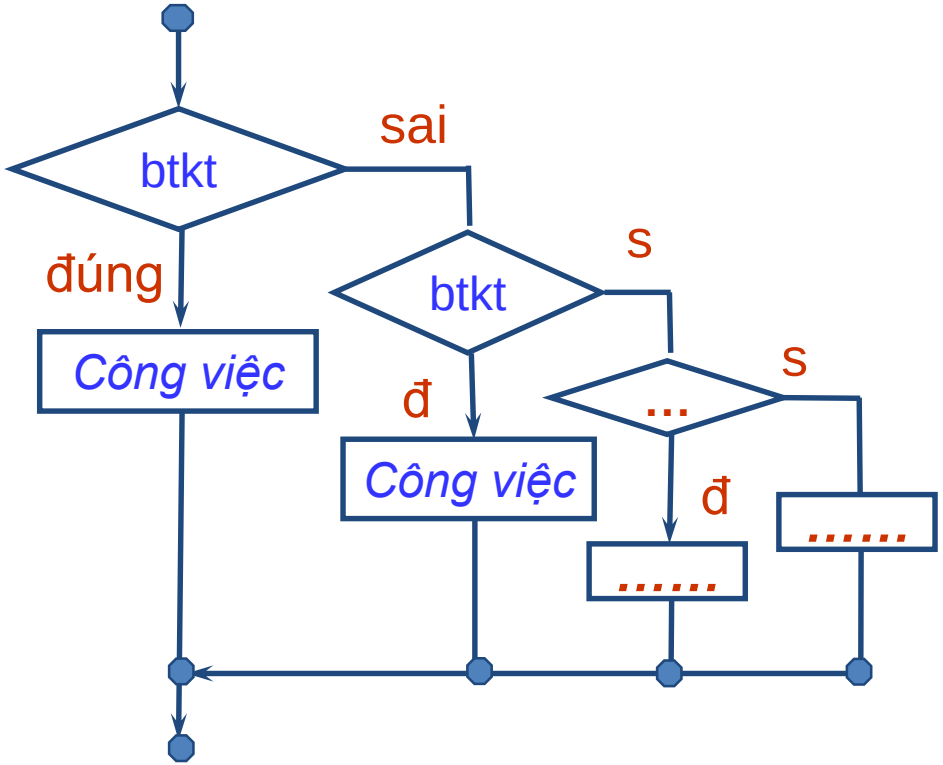
Nếu a<b thì thông báo a<b, ngược lại thông báo b<a .

```
if(a<b)
    cout<<"a<b";
else
    cout<<"b<a";
```




# Cấu trúc rẽ nhánh if dạng 2

- Cấu trúc rẽ nhánh if…else có thể lồng nhau nhiều cấp:




# Cấu trúc rẽ nhánh if dạng 2

- Ví dụ rẽ nhánh lồng nhiều cấp

  Nếu điểm TB>=8.5 thì xếp loại Giỏi

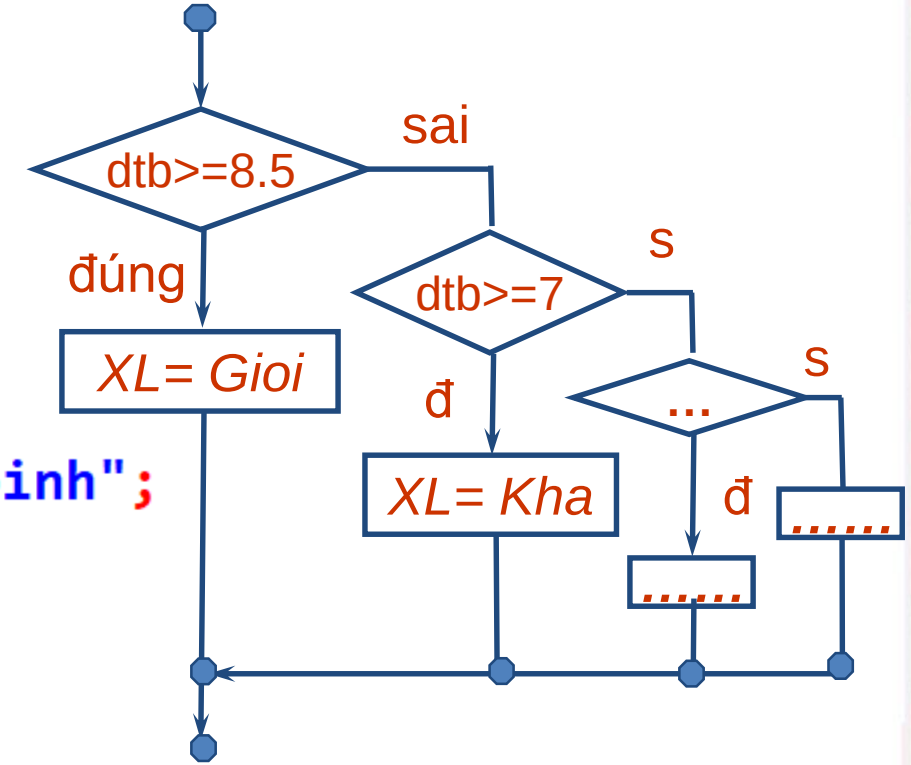
  Ngược lại, nếu điểm TB>=7 thì xếp loại Khá

  Ngược lại, nếu điểm tb>=5.5 thì xếp loại TB

  Còn lại, xếp loại Yếu

# Cấu trúc rẽ nhánh if dạng 2

```
if(dtb>=8.5)
    cout<<"xep loai gioi";
else if(dtb>=7)
    cout<<"xep loai kha";
else if(dtb>=5.5)
    cout<<"xep loai trung binh";
else
    cout<<"xep loai yeu";
```




# Cấu trúc rẽ nhánh if

- Chú ý:
  - *Biểu thức điều kiện* luôn được đặt trong ngoặc ( )
  - Trong biểu thức điều kiện phức tạp cần sử dụng các toán tử logic &&, ||, ! để kết hợp các biểu thức điều kiện đơn
  - Nếu <*công việc*> có từ 2 lệnh trở lên thì nhất thiết phải đặt vào cặp { }

# Câu lệnh rẽ nhánh **switch**

- Là một câu lệnh lựa chọn đa hướng
- Cú pháp:

```
switch (<biểu thức>)
{
    case gt_1 : congviec1; break;
    case gt_2 : congviec2; break;
    ........
    case gt_N : congviecN; break;
    default: congviecN+1;
}
```

# Câu lệnh rẽ nhánh **switch**

- Ý nghĩa: tương ứng với từng giá trị **gt_i** của **<biểu thức>** mà **<công việc i>** được thực hiện. Trong trường hợp giá trị của biểu thức không trùng với giá trị **gt_i** nào, thì **<công việc thứ N+1>** được thực hiện.

# Câu lệnh rẽ nhánh **switch**

- Chú ý
  - **<biểu thức>** phải có giá trị trả về thuộc kiểu số nguyên (**int, longint, shortint**) hoặc kiểu ký tự (**char**) hoặc kiểu logic (**bool**)
  - Các **gt_i** phải cùng kiểu với **<biểu thức>**
  - Thường thì **<biểu thức>** là tên các biến, **gt_i** giá trị có thể xảy ra của biến đó.

# Câu lệnh rẽ nhánh **switch**

- Chú ý
  - Sau mỗi **<công việc>** phải có lệnh **break** để thoát khỏi **switch**, nếu không thì các công việc sau nó đều được thực hiện cho đến khi gặp lệnh **break** khác.

# Câu lệnh rẽ nhánh **switch**

- Chú ý
  - Nếu nhiều giá trị của biểu thức cùng thực hiện một công việc thì có thể viết

```
switch (<biểu thức>)
{
    case gt_1:
    case gt_2:
    ……
    case gt_k: congviec; break;
    ……
    case gt_N : congviecN; break;
    default: congviecN+1;
}
```

# Ví dụ lệnh switch

VD1: Kiểm tra một số. Cho biết số đó là tháng mấy.

Thông báo nếu số đó không phải là số của một tháng

```
switch(t)
{
    case 1:
        cout<<"Ban vua nhap thang mot ";
        break;
    case 2:
        cout<<"Ban vua nhap thang hai ";
        break;
    ........
    default:
        cout<<"Ban da nhap sai.";
}
```

# Ví dụ lệnh switch

VD2: Cho biết ký tự vừa nhập là nguyên âm hay phụ âm

```
char c;
cout<<"Nhap mot chu cai thuong:";  cin>>c;
if(c<'a' || c>'z')
    cout<<"Ban da khong nhap dung yeu cau!";
else
    switch(c)
    {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        cout<<"Chu cai vua nhap la nguyen am.";  break;
        default:
        cout<<"Chu cai vua nhap la phu am." ;
    }
```


CẤU TRÚC LẶP

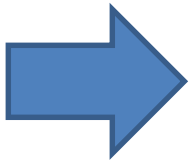
# Xét bài toán 1

- Viết chương trình xuất ra màn hình 10 số tự nhiên đầu tiên, mỗi số cách nhau một dấu tab

Viết sao đây ta????

```
cout<<0<<"\t";
cout<<1<<"\t";
cout<<2<<"\t";
cout<<3<<"\t";
cout<<4<<"\t";
cout<<5<<"\t";
cout<<6<<"\t";
cout<<7<<"\t";
cout<<8<<"\t";
cout<<9<<"\t";
```

```
cout<<0<<"\t";
cout<<1<<"\t";
cout<<2<<"\t";
cout<<3<<"\t";
cout<<4<<"\t";
cout<<5<<"\t";
cout<<6<<"\t";
cout<<7<<"\t";
cout<<8<<"\t";
cout<<9<<"\t";
```



```
int i = 0;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
```

- Như vậy, 2 lệnh

```
cout<<i<<"\t"; i++;
```

được viết đi viết lại 10 lần.

=> Lặp có biết trước số lần

```
int i = 0;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
cout<<i<<"\t"; i++;
```

# Xét bài toán 2

- Viết chương trình xuất ra các chữ số của một số nguyên dương bất kỳ nhập từ bàn phím, mỗi chữ số cách nhau một dấu tab.

Viết sao đây ta????

# Giải pháp

- Vì không biết số đó có bao nhiêu chữ số
- Tôi sẽ lấy ra chữ số cuối cùng của số đó
- Sau đó giảm số đó đi 10 lần
- Cứ làm như vậy cho đến khi số đó bị giảm về 0

=> lặp không biết trước số lần lặp

# Khái niệm cấu trúc lặp

- Là cấu trúc mà trong đó, có một số lệnh được thực hiện lặp đi lặp lại cho đến khi một điều kiện dừng nào đó thỏa mãn
- Có 3 cấu trúc lặp:
  - **for**
  - **while**
  - **do…while**

# Cấu trúc lặp **for**

- Là cấu trúc lặp khi biết trước số lần lặp
- Cú pháp:

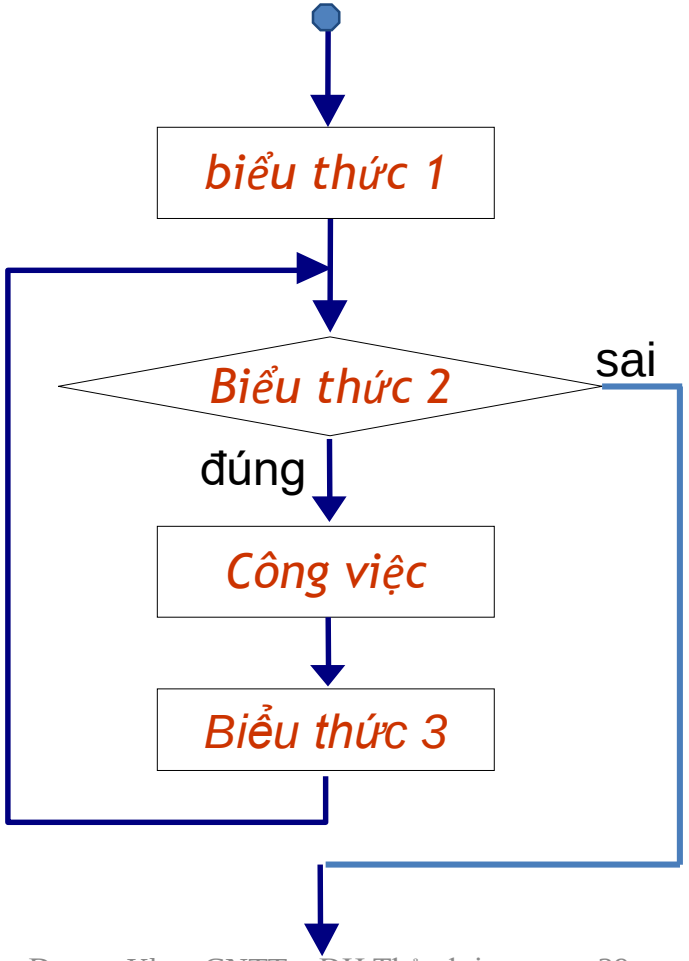
```
for (Biểu thức 1; Biểu thức 2; Biểu thức 3)
    <công việc>
```

# Cấu trúc lặp **for**

- Biểu diễn vòng lặp for bằng sơ đồ khối

**`for`** (***Biểu thức 1; Biểu thức 2; Biểu thức 3***)

*<công việc>*




# Cấu trúc lặp **for**

- Trong đó:
- ***Biểu thức 1:*** thường là biểu thức khai báo hoặc khởi tạo biến chạy (biến điều khiển vòng lặp)
- ***Biểu thức 2 (điều kiện dừng):*** là biểu thức logic kiểm tra biến chạy. Vòng lặp sẽ dừng khi biểu thức này sai.

# Cấu trúc lặp **for**

- Trong đó:
- ***Biểu thức 3:*** là một biểu thức làm thay đổi giá trị của biến chạy với mục đích làm cho biểu thức 2 có giá trị sai, nếu không vòng lặp sẽ bị chạy vô hạn.
- Dấu phân cách giữa các biểu thức phải là dấu **;**
- Công việc có thể là một lệnh hoặc một khối lệnh

# Ví dụ cấu trúc lặp **for**

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

Trong đó:

- ***Biểu thức 1***: vừa khai báo, vừa khởi tạo i = 0
- ***Biểu thức 2***: điều kiện dừng, i còn nhỏ hơn 10 thì còn làm
- ***Biểu thức 3***: thay đổi biến i tăng dần lên 10
- ***Công việc*** có 1 lệnh nên không cần cặp ngoặc **{ }**

# Câu hỏi

- Hãy cho biết kết quả của đoạn chương trình sau:

```
for(char i = 100; i<110; i++);
    cout<<i<<" ";
```

a) Chương trình báo lỗi

b) In ra màn hình:

```
100 101 102 103 104 105 106 107 108 109
```

c) In ra màn hình:

```
d e f g h i j k l m
```

# Câu hỏi

- Hãy cho biết kết quả của đoạn chương trình sau:

```
int i;
for(i = 1; i<10; i++);
{   cout<<i<<" "; cout<<endl;}
```

c) In ra màn hình:

```
1 2 3 4 5 6 7 8 9
```

b) In ra màn hình:

```
10
```

```
1
2
3
4
5
6
7
8
9
```

# Bài tập

1. Viết chương trình nhập vào 2 số a, b.

- Tính tổng các số chia hết cho 3 trong khoảng a,b.
- In ra màn hình các số nguyên liên tiếp từ a đến b theo thứ tự tăng dần và giảm dần bằng các cách có thể
- Tính tổng các số nguyên liên tiếp từ a đến b

2. Viết các chương trình tính các giai thừa của số tự nhiên N theo các công thức sau (N nhập vào từ bàn phím)

a) Giai thừa của N: N!= 1.2.3…N

b) Giai thừa lẻ của N: (2N+1)!! = 1.3.5…(2N+1)

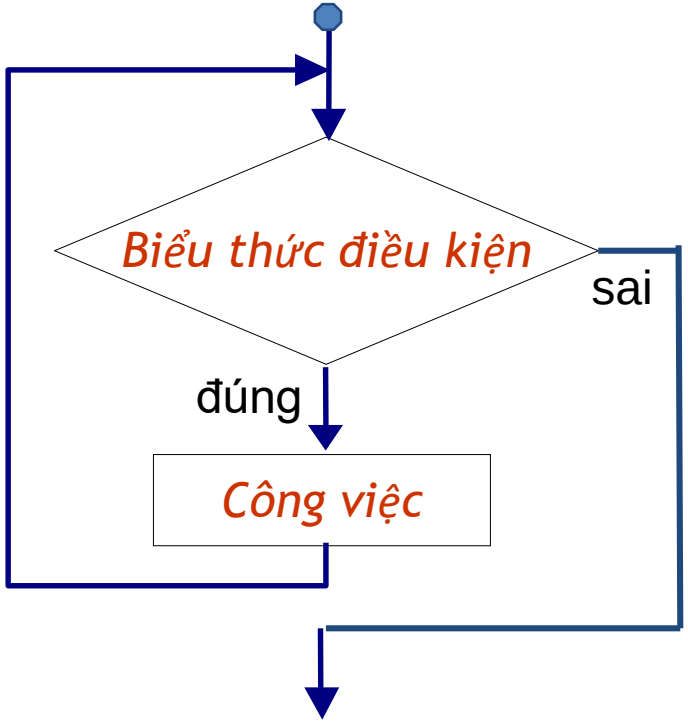
c) Giai thừa chẵn của N: (2N)!! = 2.4.5…(2N)

# Cấu trúc lặp **while**

- Là cấu trúc lặp khi không biết trước số lần lặp
- Cú pháp:

**while ( biểu thức điều kiện)**

**<công việc>**




# Cấu trúc lặp **while**

- Trong đó:
- ***Biểu thức điều kiện:*** là biểu thức xác định điều kiện dừng cho vòng lặp.
  - Nếu biểu thức còn trả ra kết quả đúng thì các công việc còn được thực hiện tiếp.
  - Nếu biểu thức trả ra kết quả sai thì vòng lặp sẽ dừng

# Cấu trúc lặp **while**

- Trong đó:
- ***Công việc:*** là một lệnh đơn hoặc là một khối lệnh nằm trong cặp ngoặc **{ }**
- *Chú ý:* trong vòng lặp phải có lệnh làm thay đổi giá trị trả về của biểu thức điều kiện nhằm hướng tới biểu thức điều kiện trở thành sai.

# Ví dụ cấu trúc lặp **while**

- Xuất ra các chữ số của một số nguyên dương nhập từ bàn phím

```
int i;
cout<<"i = "; cin>>i;
while(i>0)
{
    cout<<i%10<<"\t";
    i /=10;
}
```

# Cấu trúc lặp **do … while**

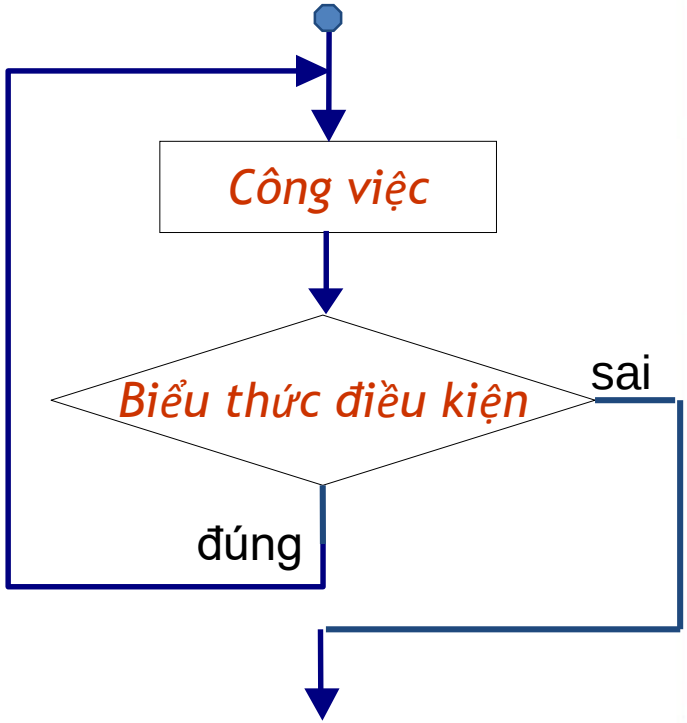
- Là cấu trúc lặp không biết trước số lần lặp
- Nhưng công việc được thực hiện ít nhất một lần trước khi lặp
- Cú pháp:

```
do
        <công việc>
while ( biểu thức điều kiện);
```

Chú ý: Phải có dấu ; ở đây

# Cấu trúc lặp **do … while**

- Sơ đồ khối của vòng lặp do…while
- Trong đó:
- ***Công việc:*** là một lệnh đơn hoặc là một khối lệnh nằm trong cặp ngoặc **{ }**




# Cấu trúc lặp **do…while**

- Trong đó:
- ***Biểu thức điều kiện:*** là biểu thức xác định điều kiện dừng cho vòng lặp.
  - Nếu biểu thức còn trả ra kết quả đúng thì các công việc lại được thực hiện tiếp.
  - Nếu biểu thức trả ra kết quả sai thì vòng lặp sẽ dừng

# Cấu trúc lặp **do…while**

- *Chú ý:* trong vòng lặp phải có lệnh làm thay đổi giá trị trả về của biểu thức điều kiện nhằm hướng tới biểu thức điều kiện trở thành sai.

# Ví dụ cấu trúc lặp **do…while**

- Nhập số x nguyên dương từ bàn phím

=> Phải nhập cho đến bao giờ x là dương mới dừng

```
int x;
do
{
    cout<<"Nhap so nguyen duong: ";
    cin>>x;
}while (x<0);
```

# Cấu trúc lồng nhau

- Các cấu trúc điều khiển có thể lồng vào nhau
- Nhưng mỗi cấu trúc đều phải được viết theo đúng cú pháp của nó

# TÌM LỖI SAI TRONG CÁC ĐOẠN CHƯƠNG TRÌNH SAU

1

```
int i=0, j=100;
While(i<100)
{
    j-=2;
}
```

2

```
int count;
  while(count<100)
  {
     cout<<count;
  }
```


3

```
char x='Y';
while(x='Y')
{
   //...
   cout<<"Continue? (Y/N)";
   cin>>x;
}
```

# BÀI TẬP

- **Bài 1: Đoạn chương trình sau hiển thị gì trên màn hình?**

```
int a=24, b=9,t;
 while(b!=0)
  {
    t= b;
    b= a%b;
    a= t;
  }
  cout<<a;
```

# BÀI TẬP

- **Bài 2: Đoạn chương trình sau hiển thị gì trên màn hình?**

```
int n=23, x=0;
do
  {
      x = x*10;
      x = x + n%10;
     n=n/10;
  }while(n!=0);

cout<<x;
```

# BÀI TẬP

- **Bài 3: Đoạn chương trình sau hiển thị gì trên màn hình?**

```
int i=0;
  while(++i<4)
     cout<<"Hello!"<<endl;
```

- **Bài 4: Đoạn chương trình sau hiển thị gì trên màn hình?**

```
  int i=0;
  do
    cout<<"hello! "<<endl;
  while (i++<4);
```