



## Chương 4

# Cấu trúc điều khiển trong C

### Nội dung

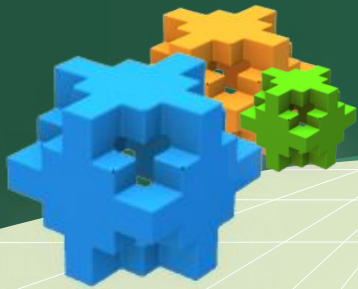
1 Cấu trúc tuần tự (sequence)

2 Câu lệnh rẽ nhánh

3 Cấu trúc lặp

4 Lệnh điều khiển





# Mục tiêu

Trình bày các cấu trúc lệnh, sinh viên hiểu và cài đặt được các cấu trúc điều khiển trong viết chương trình, bao gồm:

1. Cấu trúc rẽ nhánh: `if...else`
2. Cấu trúc lựa chọn: `switch...case`
3. Cấu trúc lặp: `while, for, do...while`
4. Lệnh điều khiển: `break, return, continue`

# 1. Các loại cấu trúc điều khiển

## TUẦN TỰ

Lệnh 1;  
Lệnh 2;  
Lệnh 3;  
....

## RỄ NHÁNH CÓ ĐIỀU KIỆN

if  
if ... else

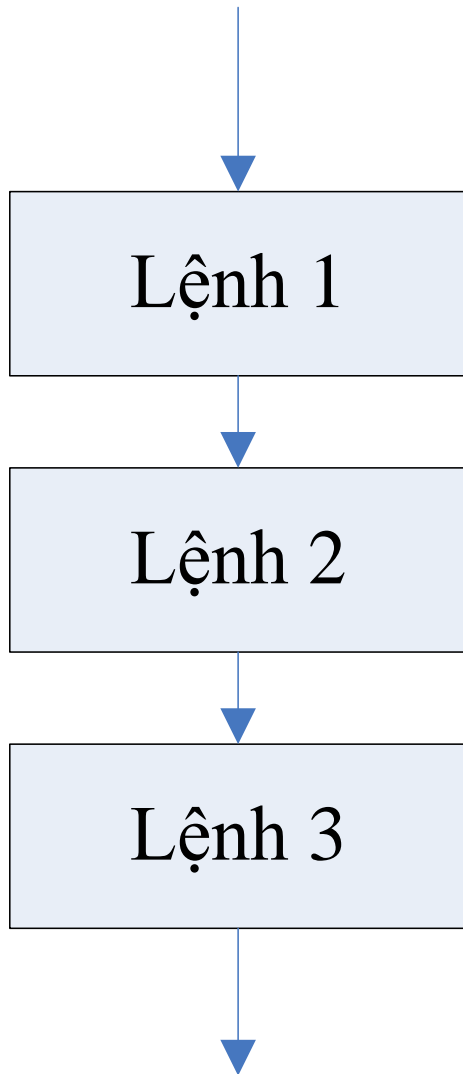
## LỰA CHỌN

switch ... case

## LẶP

for  
while  
do ... while

# 1. Cấu trúc tuần tự (sequence)



- Tuần tự thực thi tiến trình, mỗi lệnh được thực thi theo một chuỗi từ trên xuống
- Thực hiện xong lệnh này rồi chuyển xuống lệnh kế tiếp
- Mỗi lệnh **đều được thực hiện và duy nhất một lần**

# Cấu trúc tuần tự (sequence)

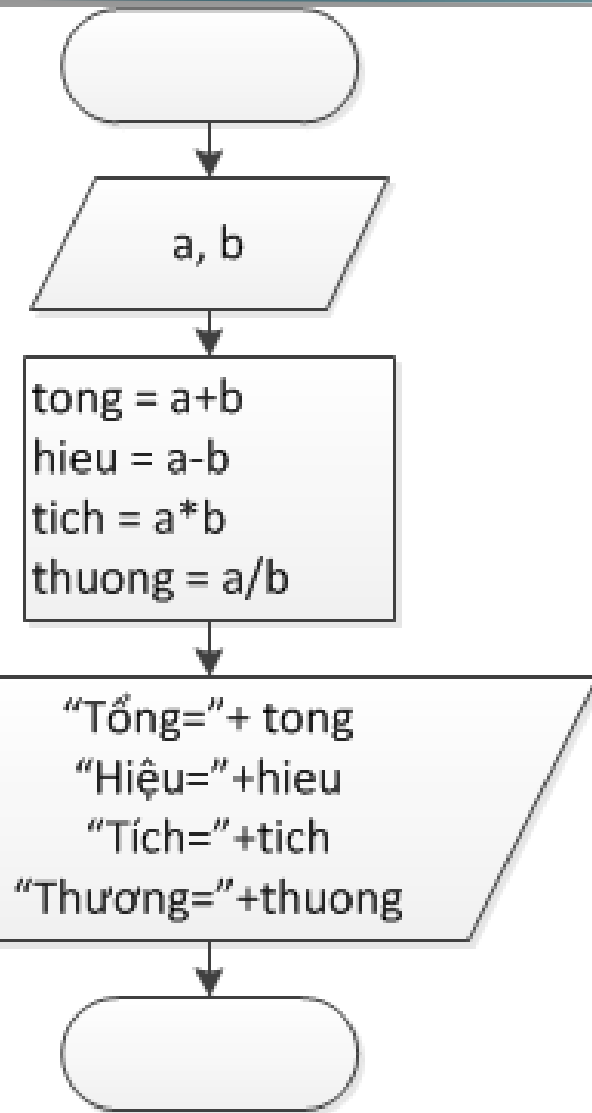
```
#include <stdio.h>
#include <conio.h>
```

```
int main()
{
    int a, b, tong, hieu, tich;
    float thuong;

    printf("Nhap vao so nguyen a: ");
    scanf("%d", &a);
    printf("Nhap vao so nguyen b: ");
    scanf("%d", &b);

    tong = a + b;
    hieu = a - b;
    tich = a * b;
    thuong = (float)a / b; //Ép kiểu

    printf("Tong: %d\n", tong);
    printf("Hieu: %d\n", hieu);
    printf("Tich: %d\n", tich);
    printf("Thuong: %f", thuong);
    getch();
    return 0;
}
```



## 2. Cấu trúc rẽ nhánh

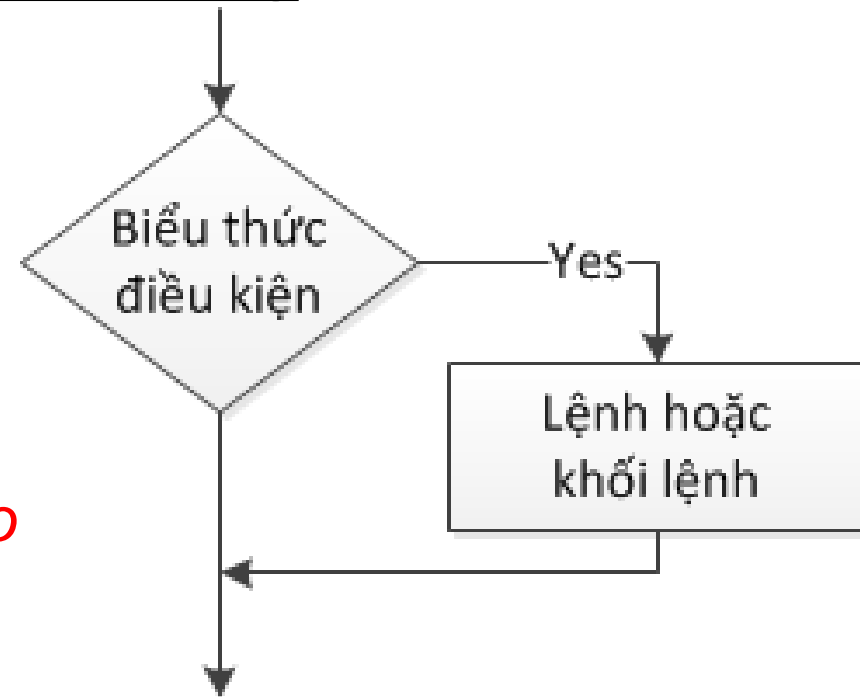
Cấu trúc rẽ nhánh chỉ cho phép thực hiện một dãy lệnh nào đó tùy thuộc vào biểu thức điều kiện

### Dạng 1: chỉ xét trường hợp đúng

*if (biểu thức điều kiện)*

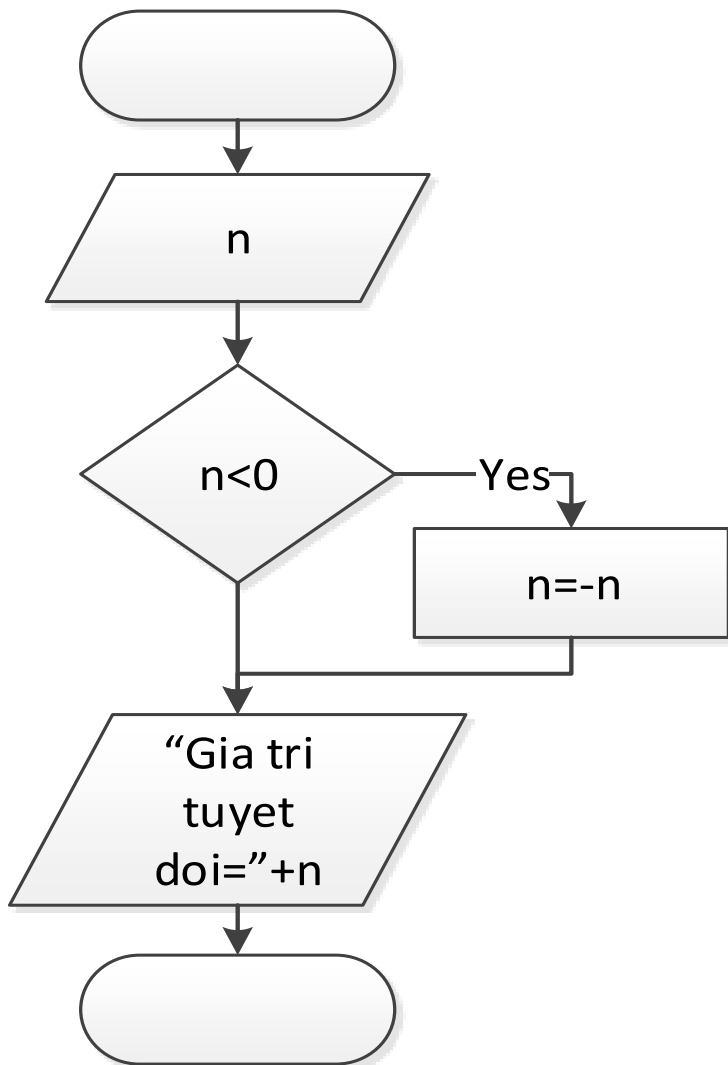
```
{  
    <khối lệnh>;  
}
```

→ Nếu biểu thức điều kiện cho khối lệnh bên trong *if*



## 2. Cấu trúc rẽ nhánh

Ví dụ: Viết chương trình nhập vào một số nguyên a, in ra giá trị tuyệt đối của a



```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout<<"Nhap vao mot so: "; cin>>a;
    if(a<0)
        a=-a;
    cout<<a<<" \n";

    return 0;
}
```

```
Nhap vao mot so: -3
3
```

## 2. Cấu trúc rẽ nhánh

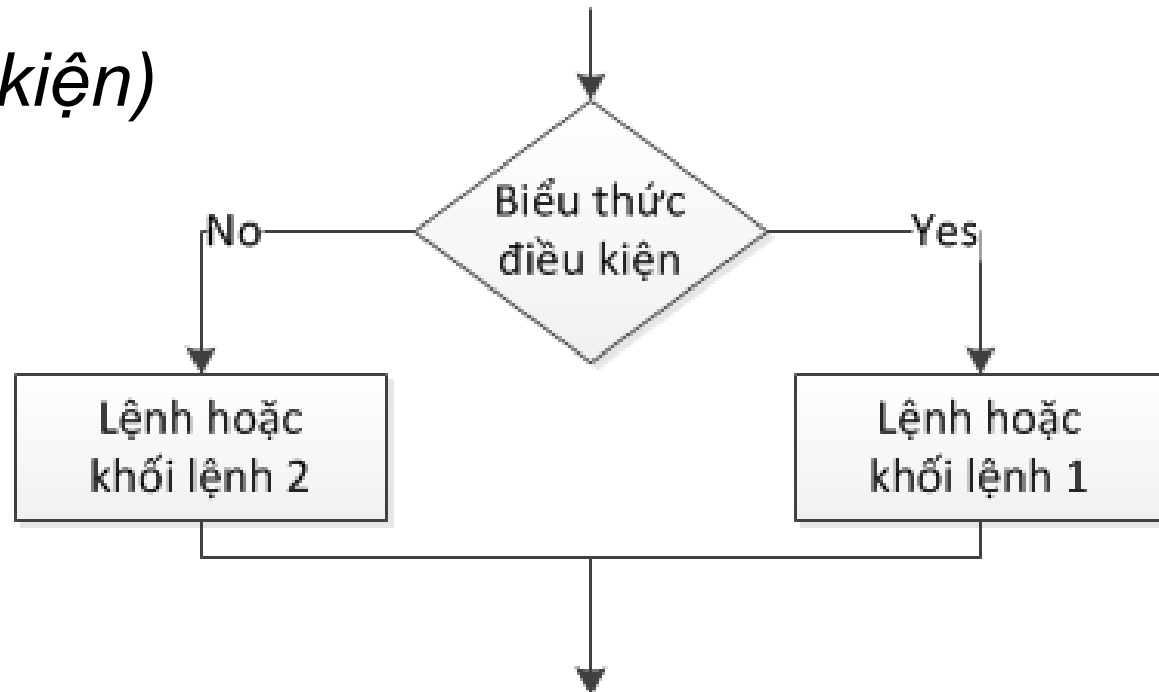
Dạng 2: xét cả hai trường hợp đúng và sai

*if* (biểu thức điều kiện)

```
{  
  <khối lệnh 1>;  
}
```

**else**

```
{  
  <khối lệnh 2>;  
}
```



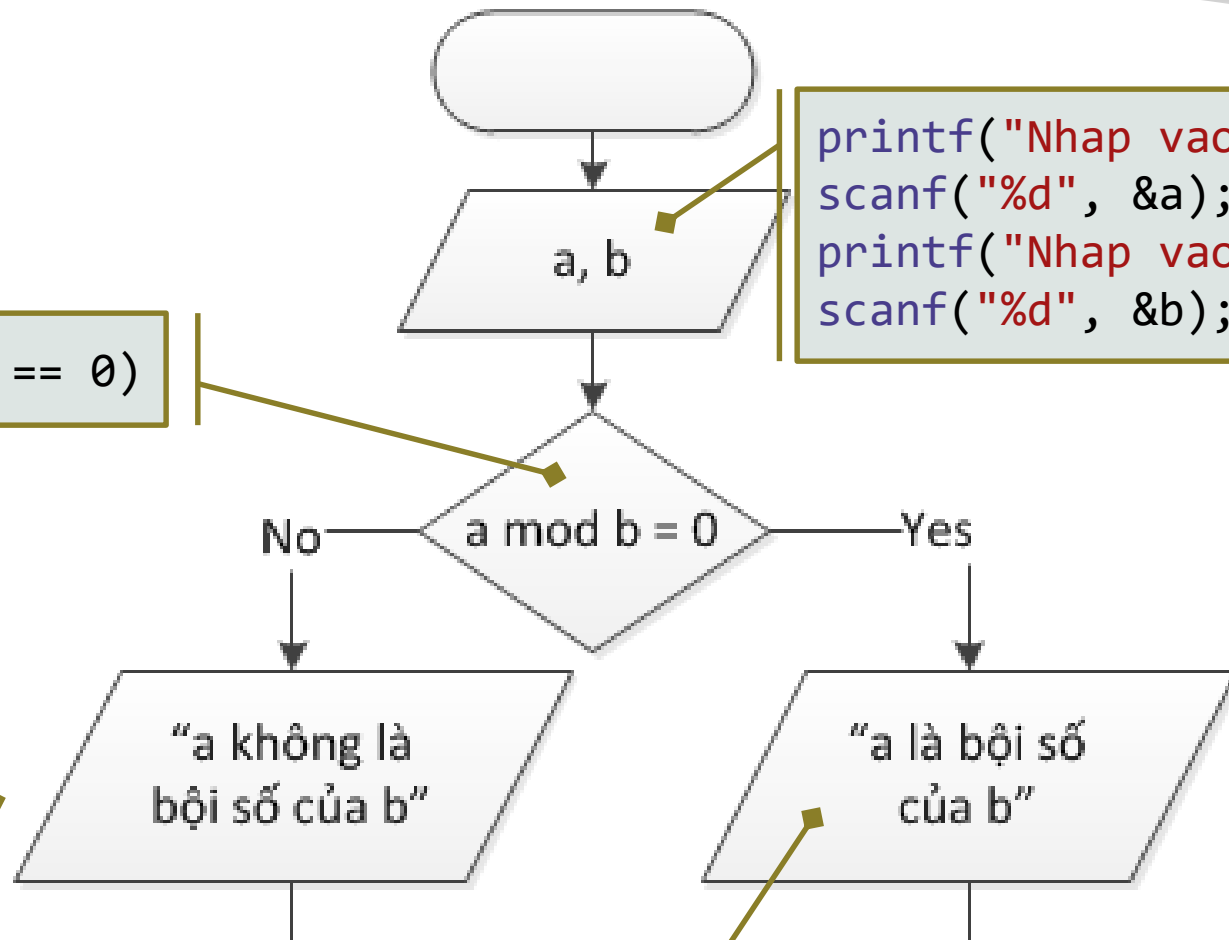
→ Nếu biểu thức điều kiện cho kết quả true thì thực hiện khối lệnh 1, ngược lại thực hiện khối lệnh thứ 2



VD: Nhập vào 2 số nguyên a và b, nếu a là bội số của b thì in “a là bội số của b”, ngược lại in “a không là bội số của b”

`if (a%b == 0)`

```
printf("Nhap vao a: ");  
scanf("%d", &a);  
printf("Nhap vao b : ");  
scanf("%d", &b);
```



“a không là  
bội số của b”

“a là bội số  
của b”

```
else  
{  
    printf("a khong la boi so cua b");  
}
```

```
printf("a la boi so cua b");
```

## 2. Cấu trúc rẽ nhánh

Ví dụ: Nhập vào một số nguyên, đưa dòng thông báo nếu là số chẵn?

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout<<"Nhập vào một số: "; cin>>a;
    if(a%2==0)
        cout<<a<<" là số chẵn \n";
    else
        cout<<a<<" là số lẻ \n";
    return 0;
}
```

```
Nhập vào một số: 6
6 là số chẵn
Press any key to continue . . .
```

## 2. Cấu trúc rẽ nhánh

Điều kiện kết hợp:

- Sử dụng các toán tử quan hệ ( $>$ ,  $<=$ ,  $>=$ ,  $==$ ,  $!=$ )
- Kết hợp với các toán tử logic ( $\&\&$ ,  $||$ ,  $!$ )

- $0 < x \leq 12$

```
if( x>0 && x<=12 )...
```

- điều kiện a, b, c là cạnh tam giác

```
if((a+b>c) && (a+c>b) && (b+c>a) && (a>0)&&(b>0)&&(c>0))
```

## 2. Cấu trúc rẽ nhánh

Điều kiện kết hợp:

```
if((n%4 ==0 && n% 100 != 0) || n %400 ==0)
    cout<<"La nam nhuan";
else
    cout<<"Khong la nam nhuan";
```

## 2. Cấu trúc rẽ nhánh

### ĐOẠN LỆNH SAU HIỂN THỊ GÌ RA MÀN HÌNH

1

Khi a có các giá trị sau: 3, 0, -5

```
if(a>0)
    b = 2*a+1;
else
    b= -2*a+1;
cout<<"gia tri b="<<b;
```

2

Khi ch có các giá trị sau: a, A, b

```
if(ch == 'a' || ch == 'b')
    cout<<"Hello!";
else
    cout<<"Goodbye!";
```

## 2. Cấu trúc rẽ nhánh

### CHƯƠNG TRÌNH SAU LÀM GÌ?

```
#include <iostream>
using namespace std;
int main()
{
    int gio; float luong;
    const float tienGio=20;
    cout<<"nhap vao so gio=";
    cin>>gio;
    if(gio>40){
        luong = tienGio * 40 + 1.5*tienGio*(gio-40);
        cout<<"So gio vuot dinh muc="<<gio-40<<endl;
    }
    else
        luong = tienGio* gio;
    cout<<"luong ="<<luong;
}
```

## 2. Cấu trúc rẽ nhánh

### Dạng 3: xét nhiều trường hợp đúng và sai

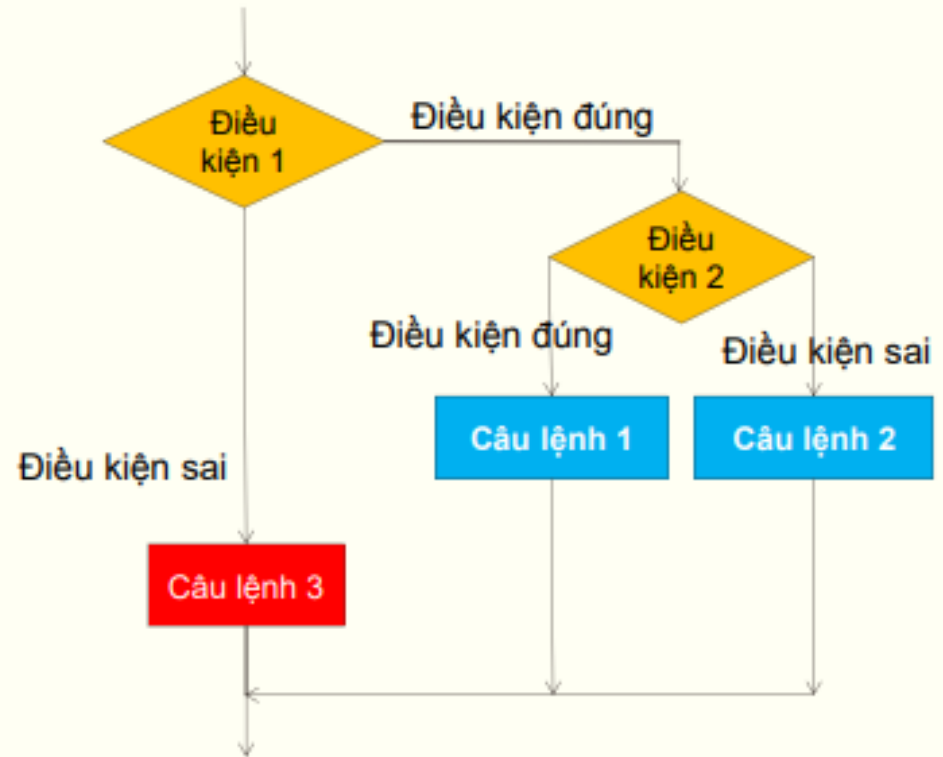
```
if(a==1)
    cout<<"One";
else if (a==2)
    cout<<"Two";
else if (a==3)
    cout<<"Three";
else if (a==4)
    cout<<"Four";
else if (a==5)
    cout<<"Five";
else if (a==6)
    cout<<"Six";
else if (a==7)
    cout<<"Seven";
else if (a==8)
    cout<<"Eight";
else if (a==9)
    cout<<"Nine";
else
    cout<<"";
```

## 2. Cấu trúc rẽ nhánh

### Dạng 3: xét nhiều trường hợp đúng và sai

#### ▪ Cú pháp:

```
if (dieukien1)
{
    if (dieukien2)
        caulenh1
    else
        caulenh2
}
else
    caulenh3
```





## 2. Cấu trúc rẽ nhánh

### Dạng 3: xét nhiều trường hợp đúng và sai

Ví dụ: Giải phương trình bậc nhất  $ax+b=0$

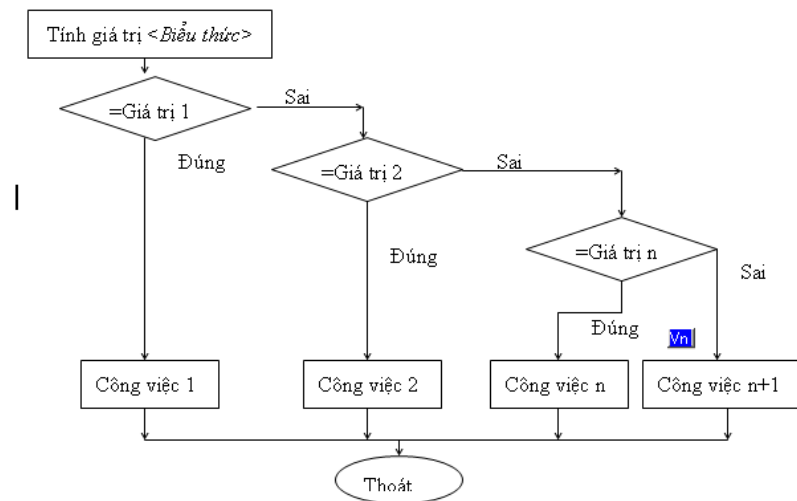
```
#include <iostream>
using namespace std;
int main()
{
    double a, b;
    cout<<"Nhap vao a, b"; cin>>a>>b;
    if(a==0)
    {
        if(b == 0)
            cout<<"Phuong trinh vo so nghiem";
        else
            cout<<"Phuong trinh vo nghiem";
    }
    else
        cout<<"Nghiem cua phuong trinh la: "<<-b/a;
    return 0;
}
```

## 2. Cấu trúc rẽ nhánh

Giải phương trình bậc hai  $ax^2 + bx + c =$

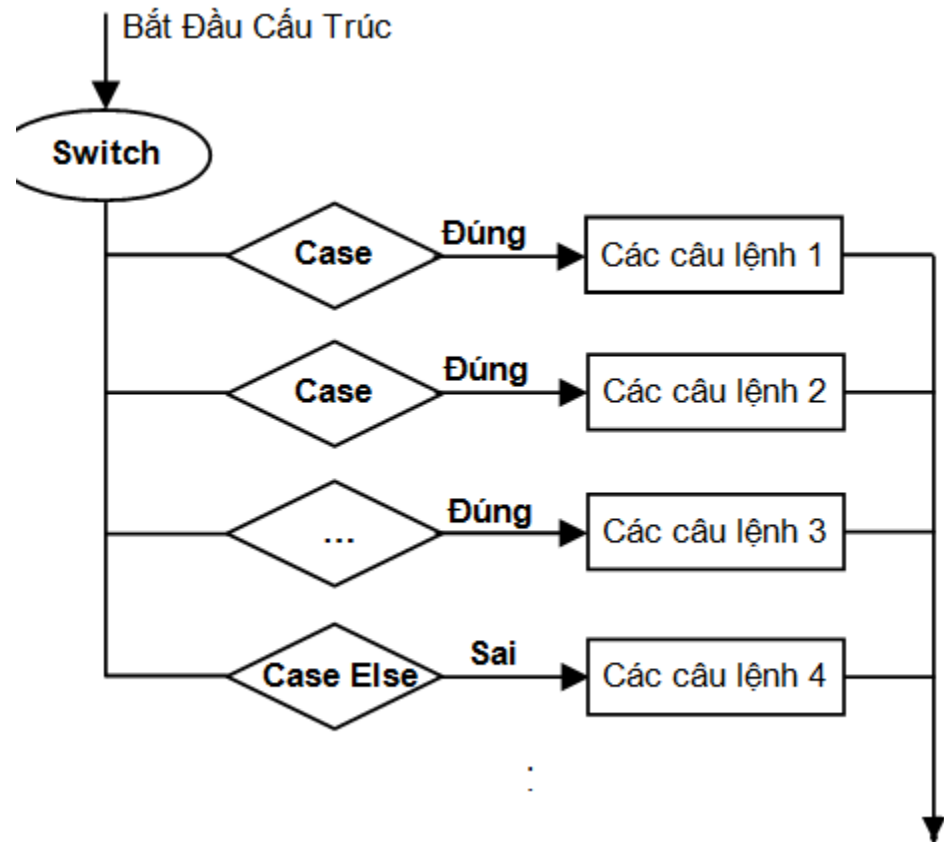
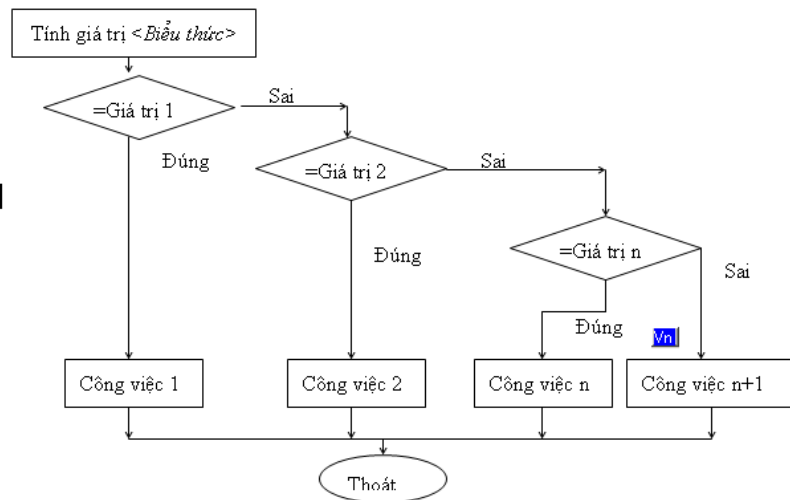
## 2. Cấu trúc rẽ nhánh

### ■ CÂU LỆNH switch



## 2. Cấu trúc rẽ nhánh

### ■ CÂU LỆNH switch



## 2. Cấu trúc rẽ nhánh

### ■ CÂU LỆNH switch

#### ■ Cú pháp:

```
switch(bieuthuc)
{
    case HANG_1:
        cau_lenh_1;
        break;
    case HANG_2:
        cau_lenh_2;
        break;
    case HANG_N:
        cau_lenh_N;
        break;
    default:
        cau_lenh_mac_dinh;
}
```

## 2. Cấu trúc rẽ nhánh

- Ví dụ:

```
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout<<"Nhap vao 2 so a va b:"; cin>>a>>b;
    char c;
    cout<<"Chon phep tinh +,-,*,/:"; cin>>c;
    switch(c)
    {
        case '+':
            cout<<a+b; break;
        case '-':
            cout<<a-b; break;
        case '*':
            cout<<a*b; break;
        case '/':
            cout<<a/b; break;
        default:
            cout<<"Ban da chon khong dung yeu cau!";
    }
}
```

### 3. Cấu trúc lặp



### 3. Cấu trúc lặp

- Cấu trúc lặp được dùng để viết chương trình nhằm thực hiện lại một khối câu lệnh cho đến khi gặp một vài điều kiện kết thúc.
  - Chẳng hạn, lặp lại các công việc tính toán cho đến khi người sử dụng nhập vào ký tự từ chối
  - Lặp lại tính toán cho đến khi số lần tính toán đạt được 100



### 3. Cấu trúc lặp

Trong ngôn ngữ C/C++ có 3 dạng câu lệnh cho cấu trúc lặp

1. Câu lệnh while
2. Lệnh do ... while
3. Lệnh for

## 3. Cấu trúc lặp

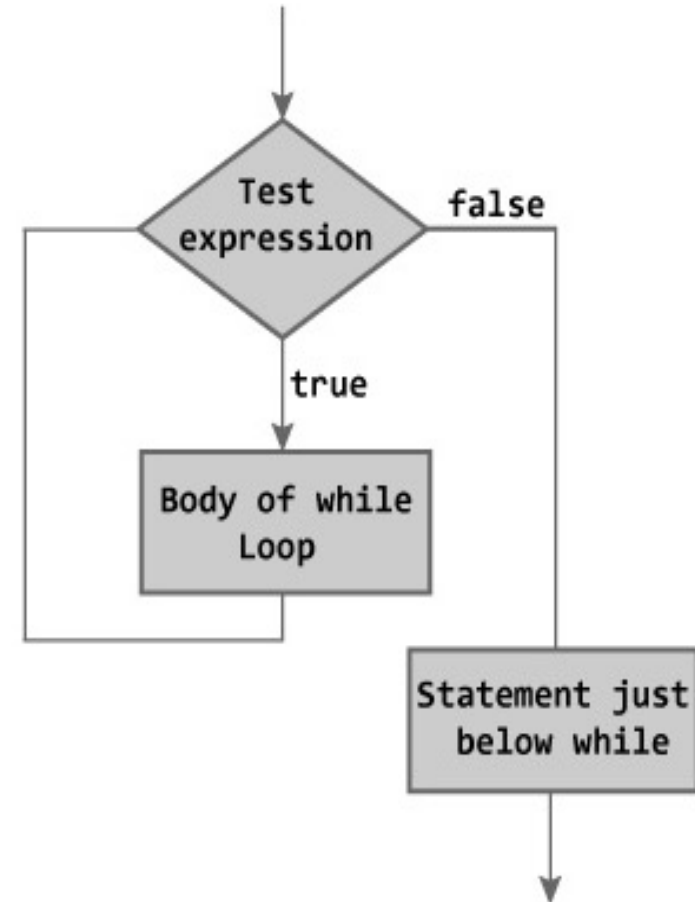
### 3.1 Lệnh while

- Câu lệnh lặp với điều kiện lặp được kiểm tra trước, chẳng hạn:
  - Trong khi **không** **mệt** thì học về lệnh while trong ngôn ngữ lập trình C/C++
  - Trong khi  $n < \text{MAX}$  thì tính toán  $a_i$  theo công thức ...

## 3. Cấu trúc lặp

### 3.1 Lệnh while

- Lưu đồ của câu lệnh while như bên cạnh.
- Ở đây điều kiện lặp được kiểm tra trước thông qua biến kiểm tra biểu thức (Test expression)
- Nếu đúng, khối các câu lệnh được lặp lại; ngược lại, thoát ra khỏi vòng lặp này



## 3. Cấu trúc lặp

### 3.1 Lệnh while

- Cú pháp câu lệnh while như sau:

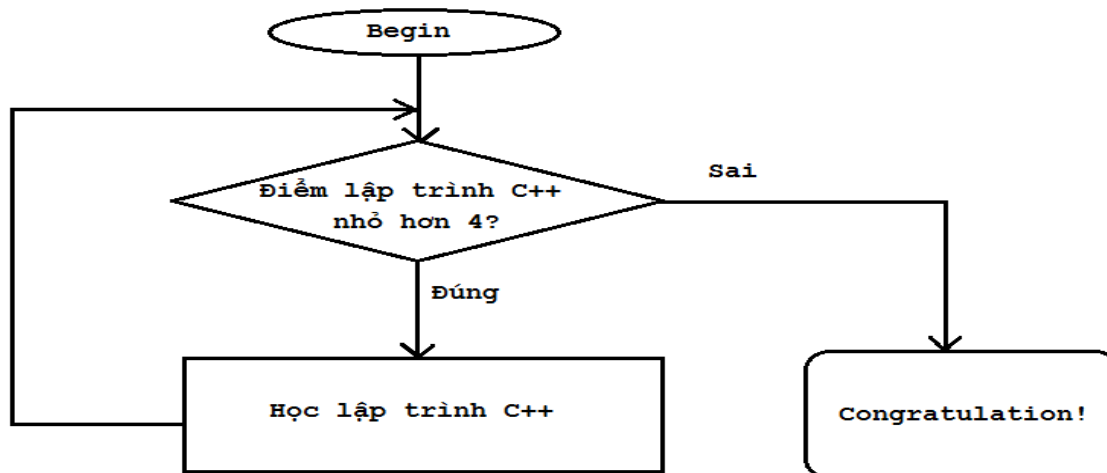
```
while( bieuThuc ){  
    /*các câu lệnh cần thực  
    hiện lặp lại */  
}
```

- Trong câu lệnh này, bieuThuc được kiểm tra trước, nếu giá trị khác không (đúng) thì những câu lệnh trong cặp {,} được thực hiện, sau đó quay trở lại kiểm tra giá trị của bieuThuc.
- Khi giá trị này bằng không (sai) thì lệnh while kết thúc.

## 3. Cấu trúc lặp

### 3.1 Lệnh while

Ví dụ: Sinh viên A đăng kí học môn lập trình C++ tại trường đại học, nếu sinh viên A không đủ điểm để qua môn học này thì sinh viên A sẽ phải học lại. Trong trường hợp sinh viên A phải học lại lần thứ 2, chúng ta lại nói rằng nếu sinh viên A không đủ điểm qua môn học này thì sinh viên A phải học lại... Vậy việc sinh viên A học lại là công việc sẽ được lặp đi lặp lại nhiều lần trong khi điều kiện sinh viên A đủ điểm để qua môn vẫn còn sai.



## 3. Cấu trúc lặp

### 3.1 Lệnh while

- Ví dụ 2: Viết chương trình nhập vào tuổi cha và tuổi con. sao nhiều năm thì tuổi cha gấp đôi tuổi con

Cha	Con	Số Năm	Cha != Con*2
41	19	0	True
42	20	1	True
43	21	2	True
44	22	3	False

## 3. Cấu trúc lặp

### 3.1 Lệnh while

Công việc cần làm:

Số năm = 0

Vòng Lặp{

Số năm ++

Tuổi cha = Tuổi cha + 1

Tuổi con = Tuổi con + 1

Nếu tuổi cha == Tuổi con \* 2

Break(Câu lệnh dùng để dừng vòng lặp bất kỳ lúc nào)

}

In ra số năm chính là kết quả mình cần

## 3. Cấu trúc lặp

### 3.1 Lệnh while

Ví dụ: Viết chương trình nhập vào số bất kỳ đến khi nhập số âm thì dừng lại

```
int main()
{
    int x = 1;

    while (x >= 0)
    {
        cout<<"Nhap gia tri bat ky, nhap -1 neu muon ngung"<<endl;
        cin>>x;
        if(x== -1)break;
    }

    printf("Chuong trinh hoan tat!\n");

    return 0;
}
```



## 3. Cấu trúc lặp

### 3.1 Lệnh while

Ví dụ: Viết chương trình nhập vào số bất kỳ đến khi nhập số âm thì dừng lại

```
int main()
{
    int x = 1;

    while (x >= 0)
    {
        cout<<"Nhap gia tri bat ky, nhap -1 neu muon ngung"<<endl;
        cin>>x;
        if(x== -1) break;
    }

    printf("Chương trình hoàn tat!\n");

    return 0;
}
```

## 3. Cấu trúc lặp

### 3.1 Lệnh while

Ví dụ: In các số từ 9 về 0 theo chiều giảm dần. Các số nằm trên 1 dòng.

```
int main()
{
    int n = 10;

    while (n-->0)
    {
        printf("%d ", n);
    }

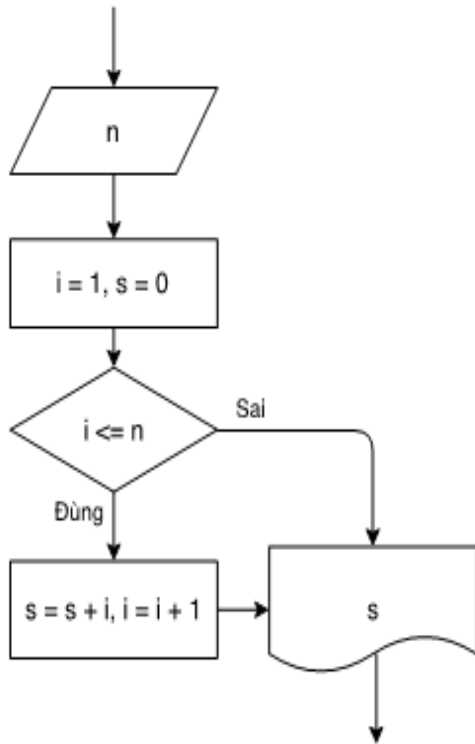
    printf("\n");

    return 0;
}
```

## 3. Cấu trúc lặp

### 3.1 Lệnh while

Ví dụ: Tính tổng của các số từ 1 đến n

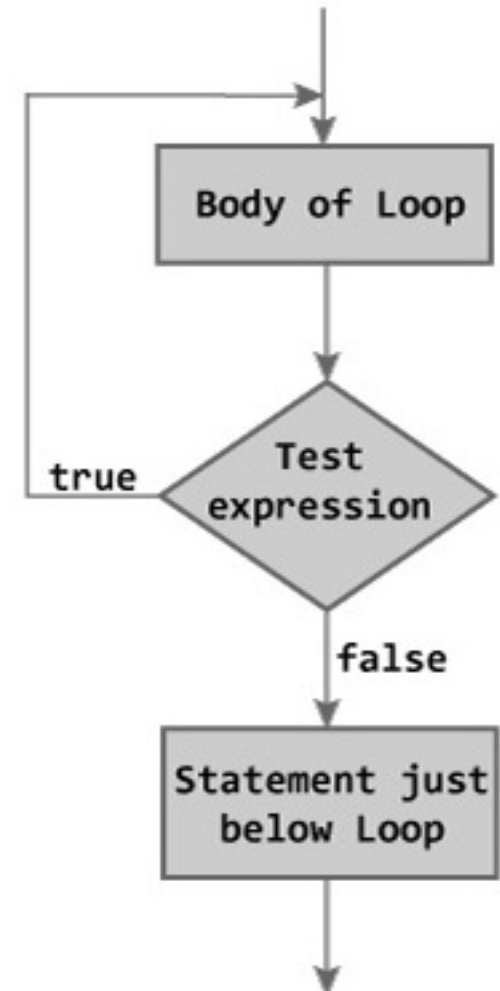


```
int n, i, s = 0;
printf( "Can tinh tong tu 1 den: " );
scanf( "%d", &n );
i = 1;
while ( i <= n ){
    s = s + i;
    i = i + 1;
}
printf( "Tong tu 1 den %d la: %d\n", n, s );
// EXIT_SUCCESS
```

## 3. Cấu trúc lặp

### 3.2 Lệnh do...while

- Lệnh do ... while khác lệnh while khi kiểm tra điều kiện lặp
- Điều kiện lặp trong lệnh do ... while được kiểm tra sau
  - Chẳng hạn, học các lệnh lặp trong ngôn ngữ lập trình C/C++ cho đến khi **mệt**.
  - Như vậy, câu lệnh lặp lại được thực hiện ít nhất là 1 lần.



## 3. Cấu trúc lặp

### 3.2 Lệnh do...while

- Cú pháp câu lệnh do ... while như sau:

```
do {
```

```
    /*các câu lệnh cần thực
```

```
    hiện lặp lại */
```

```
} while ( bieuThuc );
```

- Trong câu lệnh này, bieuThuc được kiểm tra sau khi thực hiện các câu lệnh lặp, nếu giá trị này bằng không (sai) thì lệnh do ... while kết thúc.

## 3. Cấu trúc lặp

### 3.2 Lệnh do...while

Ví dụ: kiểm tra dữ liệu nhập vào có thể là một tháng trong năm hay không

```
#include <iostream.h>

int main()
{
    int month;
    do
    {
        cout<<"Nhap vao 1 thang bat ky tu 1 den 12 trong nam 2018:";
        cin >> month;
    } while (month < 1 || month > 12);

    return 0;
}
```

## 3. Cấu trúc lặp

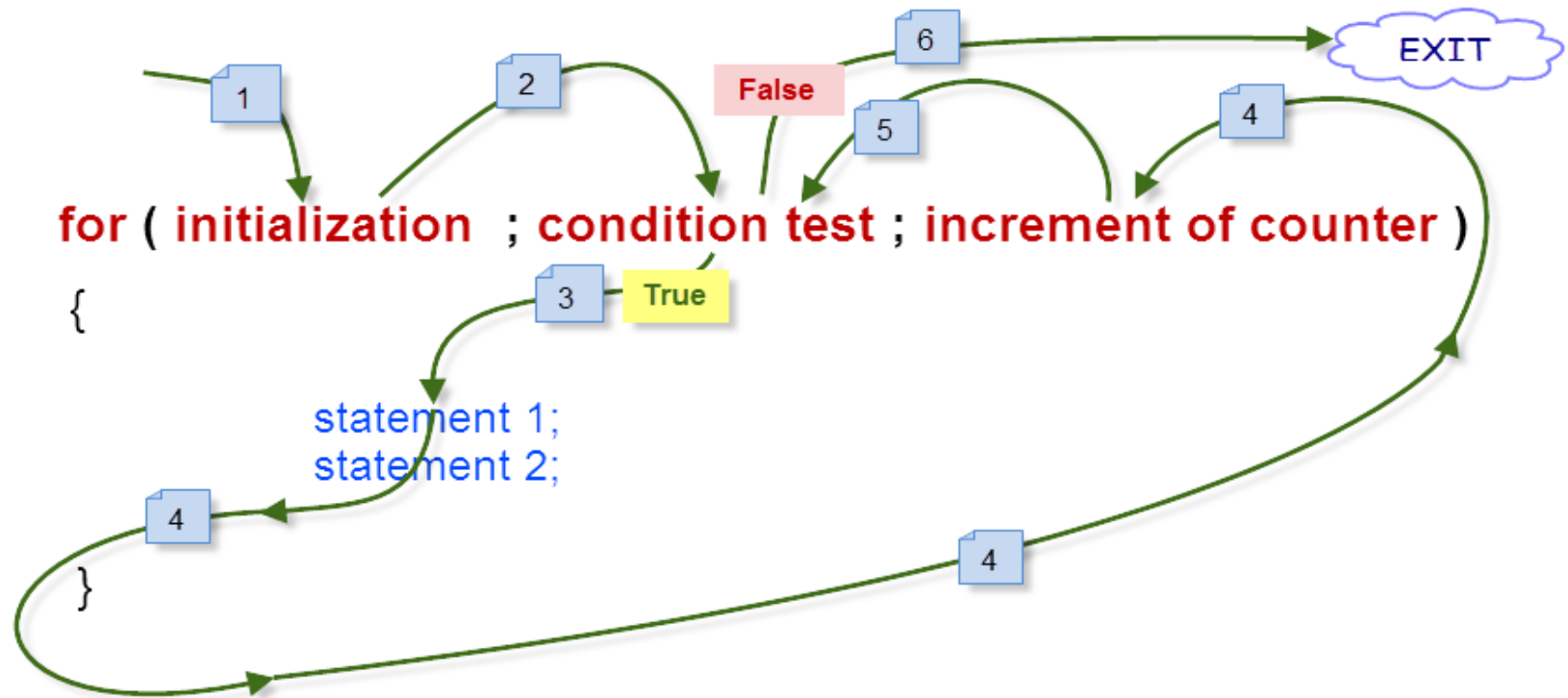
### 3.3 Lệnh for

- Cú pháp câu lệnh for như sau:

```
for( lenhKhoiDong; bieuThuc; lenhCapNhat ){  
    /*các câu lệnh cần thực  
    hiện lặp lại */  
}
```

## 3. Cấu trúc lặp

### 3.3 Lệnh for



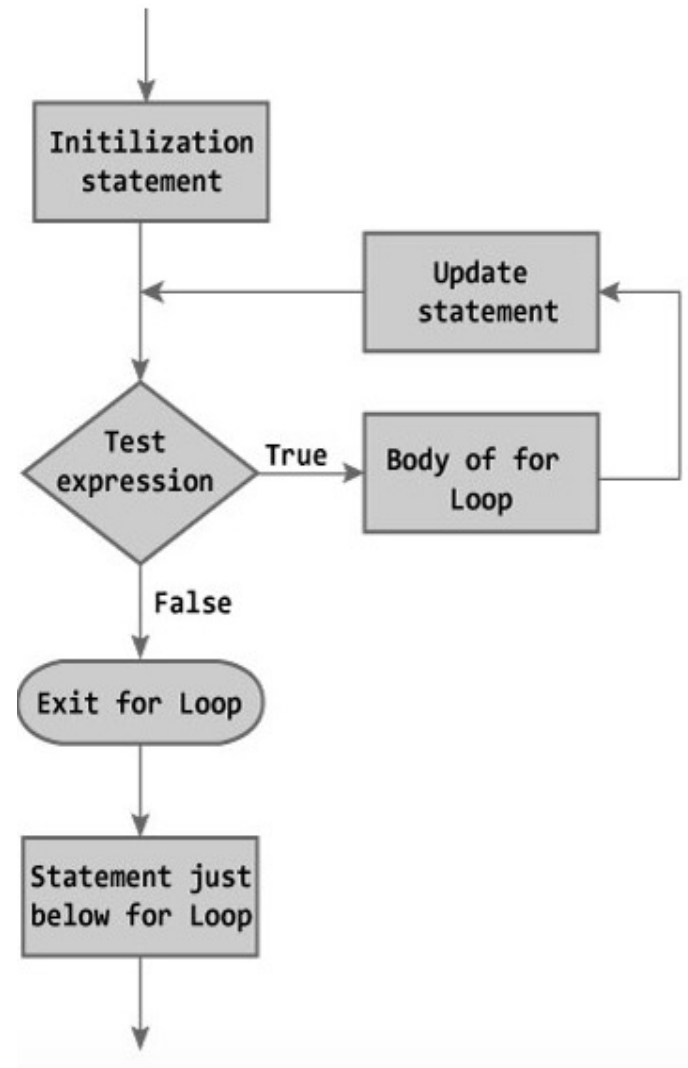


## 3. Cấu trúc lặp

### 3.3 Lệnh for

Lưu đồ của for như bên.

Trong đó, Initialization statement là lệnh khởi động, Update statement là lệnh cập nhật



## 3. Cấu trúc lặp

### 3.3 Lệnh for

- Tính tổng từ 1 đến n

```
#include <stdio.h>
int main(){
    int n, i, s = 0;
    printf( "Can tinh tong tu 1 den: " );
    scanf( "%d", &n );
    for( i = 1; i <= n; i++ )
        s += i;
    printf( "Tong tu 1 den %d la %d\n", n, s );
    return 0;
}
```

## 3. Cấu trúc lặp

### 3.3 Lệnh for

- Tính  $n!$


```
#include <stdio.h>
int main(){
    int n, num;
    long long gt;

    printf( "Nhap vao so nguyen n de tinh n!: " );
    scanf( "%d", &n );
    gt = 1; num = n;
    for( gt = 1, num = n; n > 0; --n )
        gt = gt*n;
    printf( "%d! = %ld\n", num, gt );
    return 0;
}
```

## 3. Cấu trúc lặp

### 3.4 Lệnh chuyển điều khiển

- Trong ngôn ngữ C/C++, có 3 lệnh chuyển điều khiển, đó là lệnh:
  - ✓ break
  - ✓ continue
  - ✓ goto
- Trong đó lệnh goto được khuyến cáo là ít sử dụng nhất.
- Lưu đồ của goto như hình bên



```
goto label;  
... ..  
... ..  
  
label:  
... ..  
... ..
```

The diagram illustrates the execution of a goto statement. A line from the 'goto label;' statement curves down and then right, ending in an arrow that points to the 'label:' line in the code block below.

## 3. Cấu trúc lặp

### 3.4 Lệnh chuyển điều khiển

- **Lệnh break**

- Như đã gặp trong câu lệnh switch ... case, lệnh break dùng để thoát ra khỏi một lệnh nào đó mà không thực hiện các lệnh tiếp theo.
- Chẳng hạn, trong câu lệnh switch ... case như sau:

```
switch ( n ) {  
  case 0:    m++;  
             break;  
  case 1:    m--;  
             break;  
  default:  
             m *= 2;  
}
```

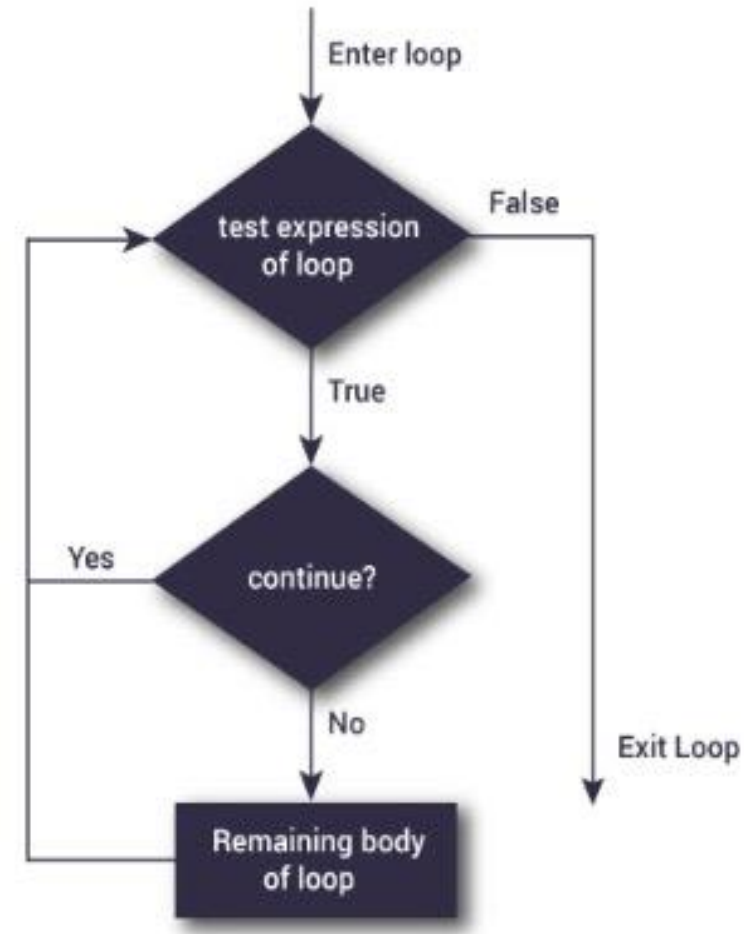
- ▶ Nếu không có lệnh break sau case 0 thì giá trị của m bị trừ 1 bởi câu lệnh m--

## 3. Cấu trúc lặp

### 3.4 Lệnh chuyển điều khiển

- **Lệnh continue**

- Nhằm bỏ qua những lệnh còn lại trong 1 lần lặp để quay trở lên lần lặp tiếp theo.
- Lưu đồ của lệnh continue như bên cạnh

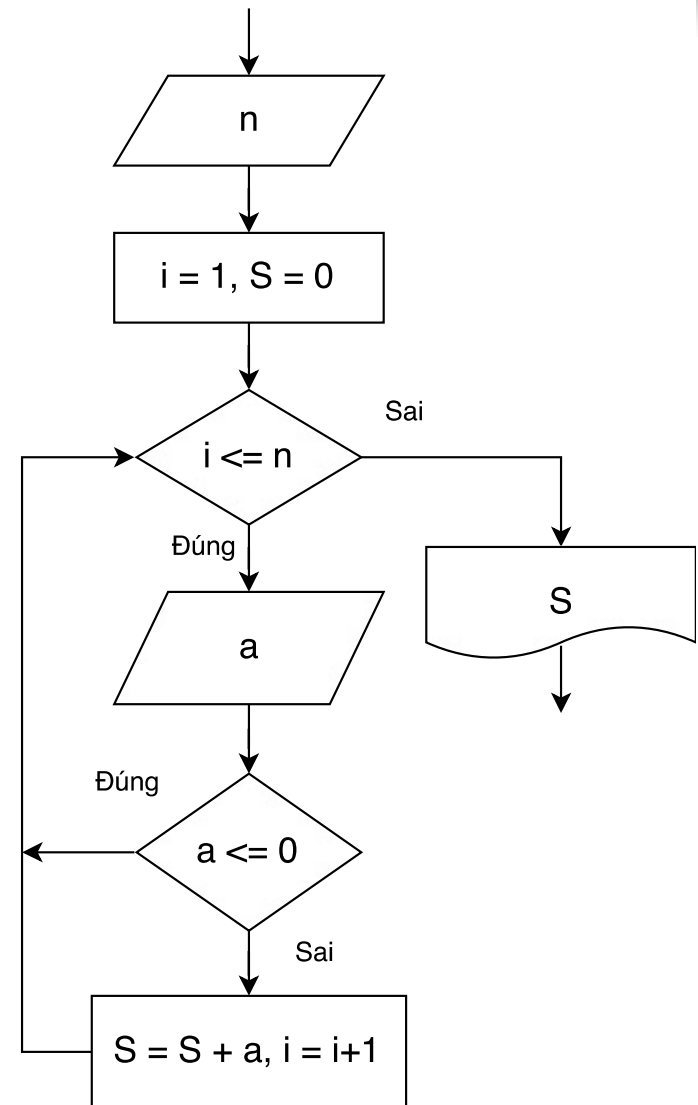


## 3. Cấu trúc lặp

### 3.4 Lệnh chuyển điều khiển

Ví dụ: tính tổng các số dương

```
int main() {  
    int n, i = 1;  
    float a, S = 0;  
    printf( "Can tinh tong cua may so: " );  
    scanf( "%d", &n );  
    while ( i <= n ){  
        printf( "Nhap so thu %d: ", i );  
        scanf( "%f", &a );  
        if ( a <= 0 )  
            continue;  
        S += a;  
        i++;  
    }  
    printf( "Tong la: %f\n", S );  
    return 0;  
}
```



# Bài tập

1. lập trình tính:

$$S1=1+2+3+4+\dots+n;$$

$$S2=1+1/2+1/3+1/4+\dots+1/n$$

2. Nhập vào số nguyên dương n và in ra màn hình

a/ các số nguyên dương từ 1 đến n

b/ tổng và trung bình cộng của n số nguyên dương này

3. Nhập vào một số tự nhiên N. Hãy cho biết

a) N có bao nhiêu chữ số

b) Tổng các chữ số của N bằng bao nhiêu



## Bài tập

4. Nhập vào n số nguyên, tính tổng các số chẵn đã nhập
5. Viết chương trình đếm và in ra số lượng các số nguyên chia hết cho 3 hoặc 7 nằm trong đoạn 1 đến 100.

# Bài tập

6. Viết chương trình nhập vào số  $n$  và in ra các ước của số  $n$  đó.
7. Viết chương trình kiểm tra 1 số có là số nguyên tố không? Số nguyên tố là số nguyên dương có duy nhất 2 ước là 1 và chính nó. Ví dụ số 2, 3, 5, ...
8. Viết chương trình kiểm tra 1 số có là số hoàn hảo không? Số hoàn hảo là số nguyên dương có tổng các ước bằng 2 lần nó. VD số 6 có các ước 1, 2, 3, 6 và tổng  $1 + 2 + 3 + 6 = 12$  (bằng 2 lần 6).

## Bài tập

9. Viết chương trình tính  $S = 1 + 1/2 + 1/3 + \dots + 1/N$
10. Viết chương trình tính tổng bình phương các số lẻ từ 1 đến n.
11. Viết chương trình tính  $n!$  biết  $n! = 1.2.3.4 \dots n$
12. Viết chương trình tính số thứ n của dãy fibonacci biết dãy  $f(n) = f(n-1) + f(n-2)$ ,  $n > 2$  và  $f(1) = 1$ ,  $f(2) = 1$ .
13. Viết chương trình nhập vào chiều dài, chiều rộng của hình chữ nhật. Vẽ hình chữ nhật dấu sao (\*) có kích thước đã nhập