

# C PROGRAMMING

(part 2)

- Cấu trúc lặp
- Hàm
- Biến toàn cục, biến cục bộ và tham số chính thức

# THÔNG TIN LIÊN HỆ

- Title: [CSLT 2020]
- Phạm Phi Nhung  
Email: [phamphinhung2898@gmail.com](mailto:phamphinhung2898@gmail.com)
- Ngô Thị Thanh  
Email: [ngothithanh2511@gmail.com](mailto:ngothithanh2511@gmail.com)

# HỆ THỐNG SAKAI

- Trang web: [learning.hvthao.com](http://learning.hvthao.com)
- User ID: MSSV - Password: MSSV
- Chọn Fundamentals of Programming - Lab

Chọn Resources

Chọn Lab

Đây là hệ thống chính để cập nhật các bài tập cũng như kiểm tra, các bạn nhớ cập nhật thường xuyên

# MỘT SỐ LƯU Ý CHO LÀM BÀI (trong trường hợp khi yêu cầu nộp bài)

Trước đầu mỗi bài làm cần có ghi chú như sau: (ví dụ)

/\*

\*MSSV: 1611xxx

\*Ho Ten: Pham Phi Nhung

\*Lop: 16TTH2

\*IDE: Microsoft Visual Studio 2015

\*/

Về yêu cầu cụ thể nộp file nào sẽ cập nhật sau

# CÁC CẤU TRÚC LẶP

-

# CÁC CẤU TRÚC LẶP\_nội dung

1. WHILE

2. FOR

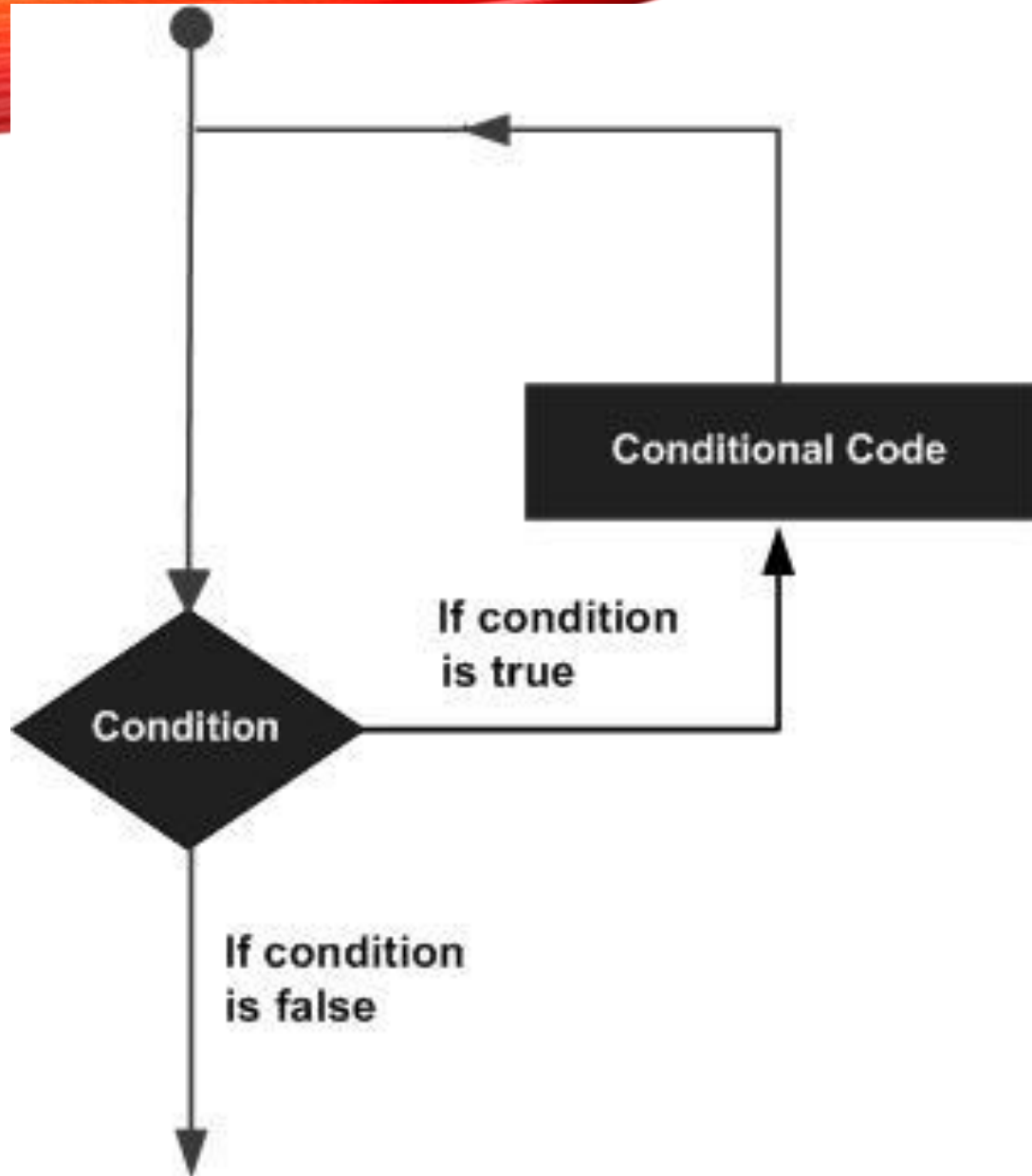
3. DO ... WHILE

4. Các lệnh điều khiển vòng lặp

Các bài tập áp dụng 3



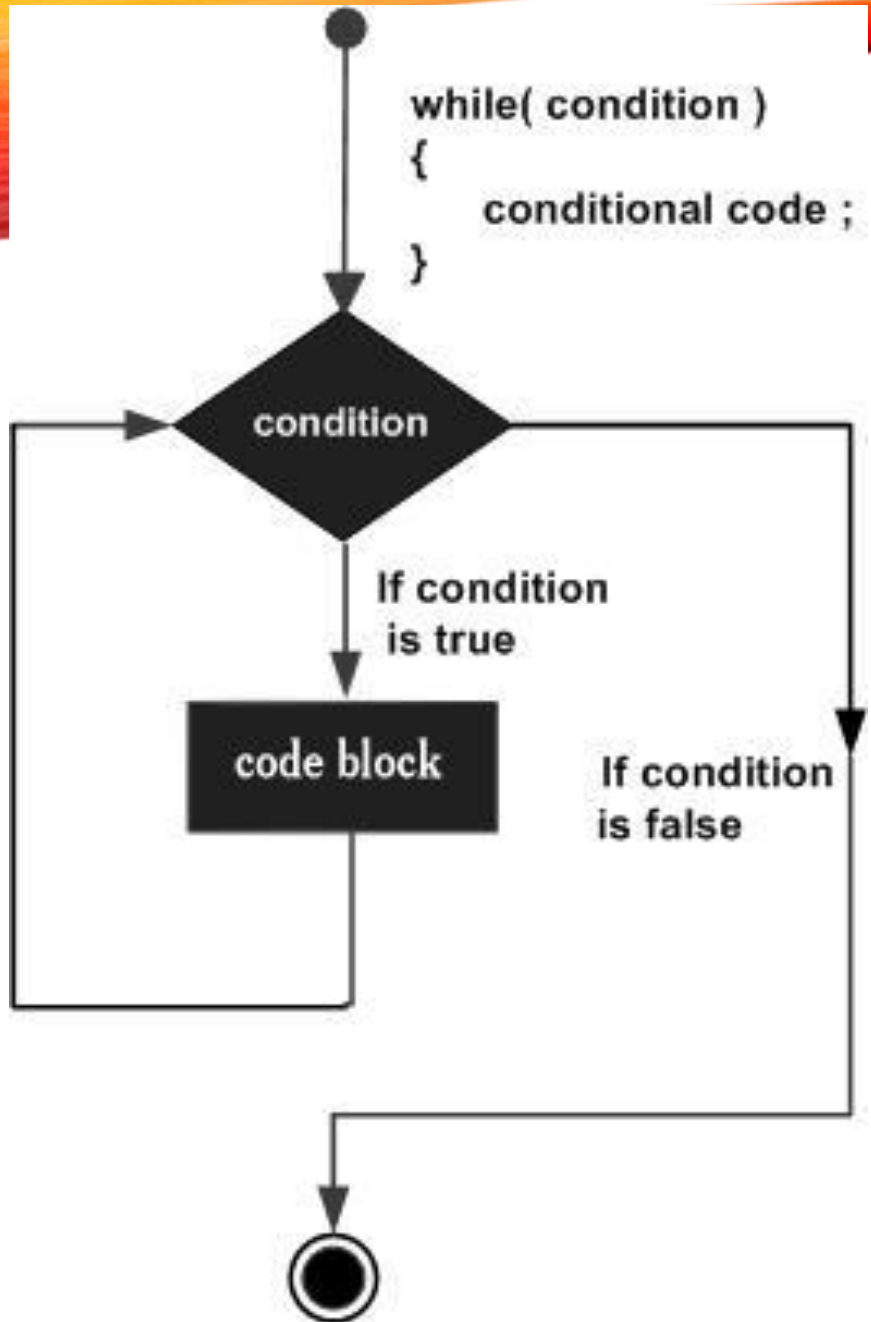
# CẤU TRÚC LẶP<sup>7</sup>



- Cho phép thực hiện một lệnh hoặc một nhóm lệnh nhiều lần
- Dạng tổng quát như hình vẽ
- Có thể sử dụng một hoặc nhiều vòng lặp trong các vòng lặp khác (lồng ghép các vòng lặp)

# WHILE

- Lặp lại một hoặc một nhóm các lệnh trong khi điều kiện đã cho là đúng
- Nó kiểm tra điều kiện trước khi thực hiện thân vòng lặp
- Thường có tên gọi khác là WHILE DO (gọi dựa theo DO...WHILE để phân biệt)

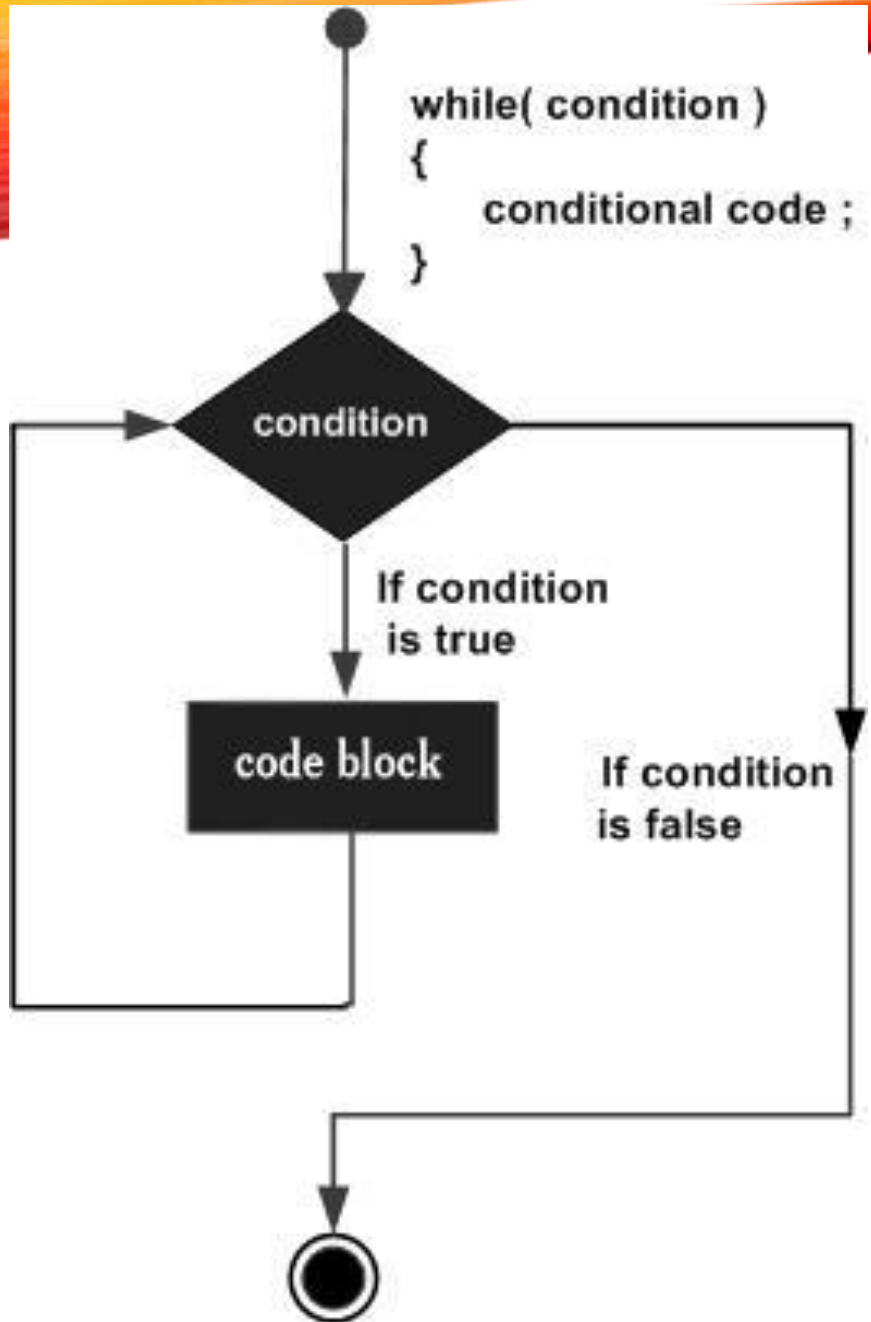




# WHILE

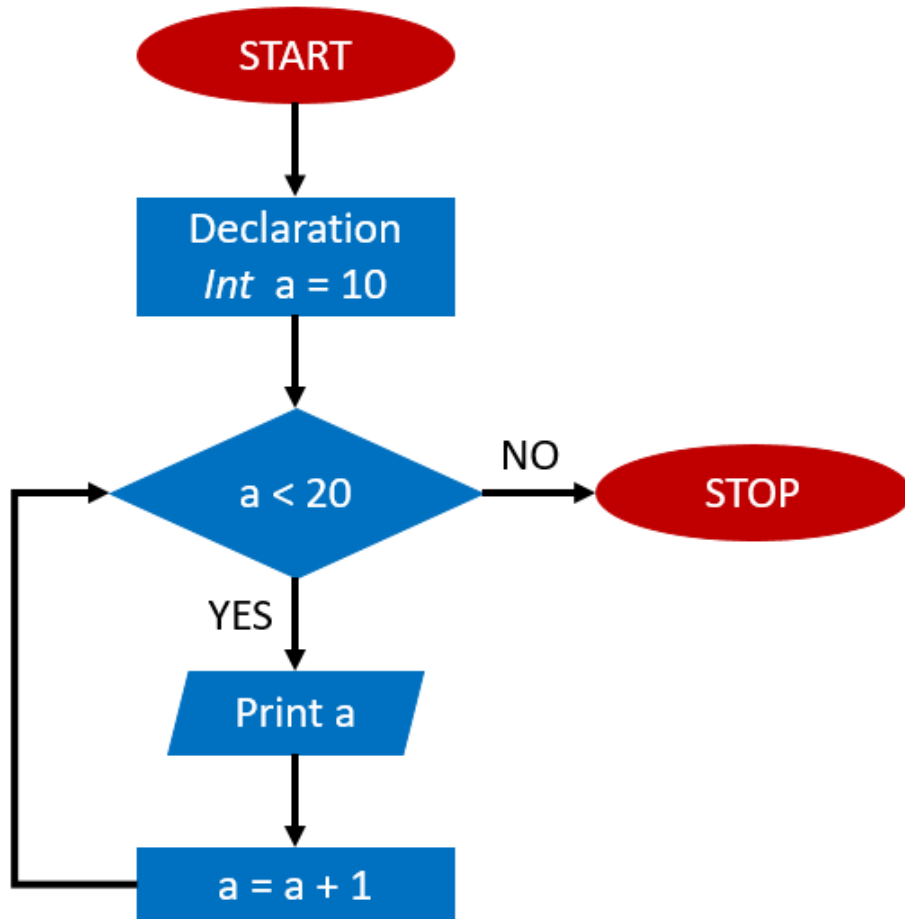
Dạng while:

```
while(dieu_kien) {  
    // dòng lệnh thực hiện  
    khi điều kiện đúng  
}
```



# WHILE\_ví dụ

10



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

```
int main () {
```

```
    /* local variable definition */
```

```
    int a = 10;
```

```
    /* while loop execution */
```

```
    while( a < 20 ) {
```

```
        printf("value of a: %d\n", a);
```

```
        a++;
```

```
    }
```

```
    return 0;
```

```
}
```

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

# WHILE\_warning/error

11

```
#include <stdio.h>
int main(){

    int n = 1;
    while(n > 10){
        printf("%d\n",n);
        n++;
    }

    int m = 1;
    while(m < 10){
        printf("%d\n",m);
        m--;
    }

    return 0;
}
```

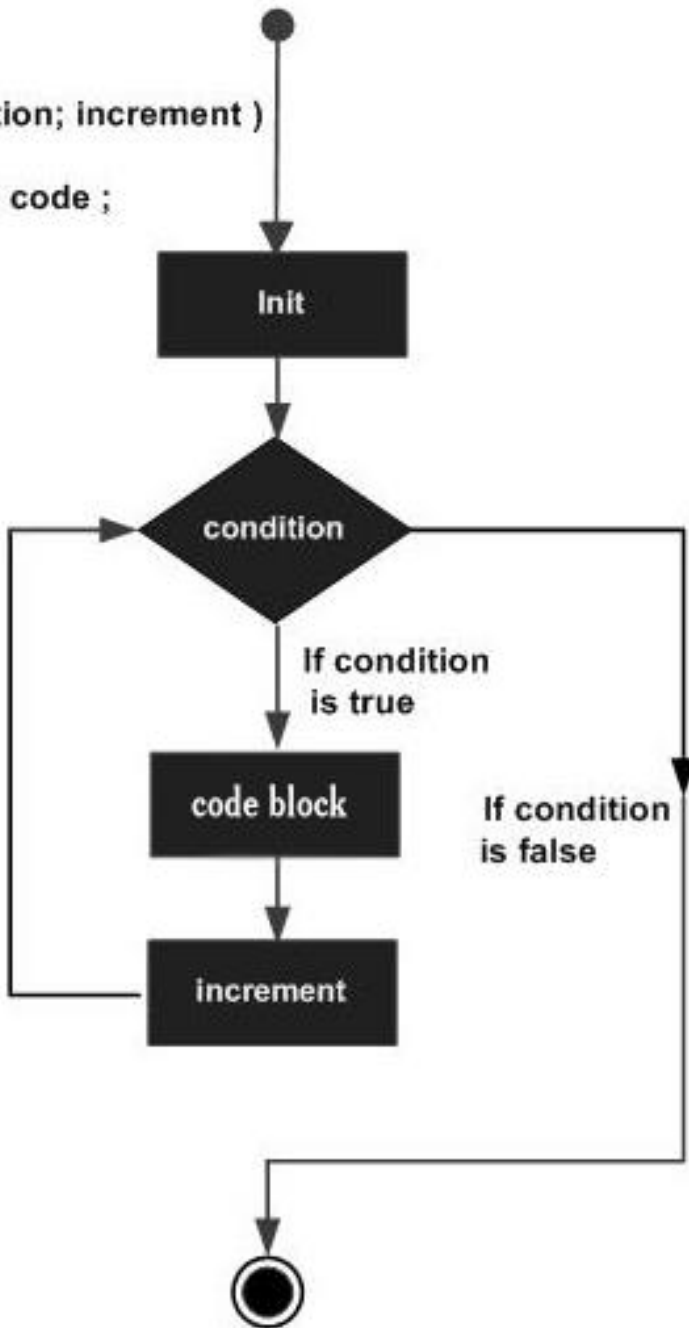
Ví dụ cụ thể thường hay mắc  
lỗi khi dùng while



bạn có tìm thấy lỗi???

# FOR

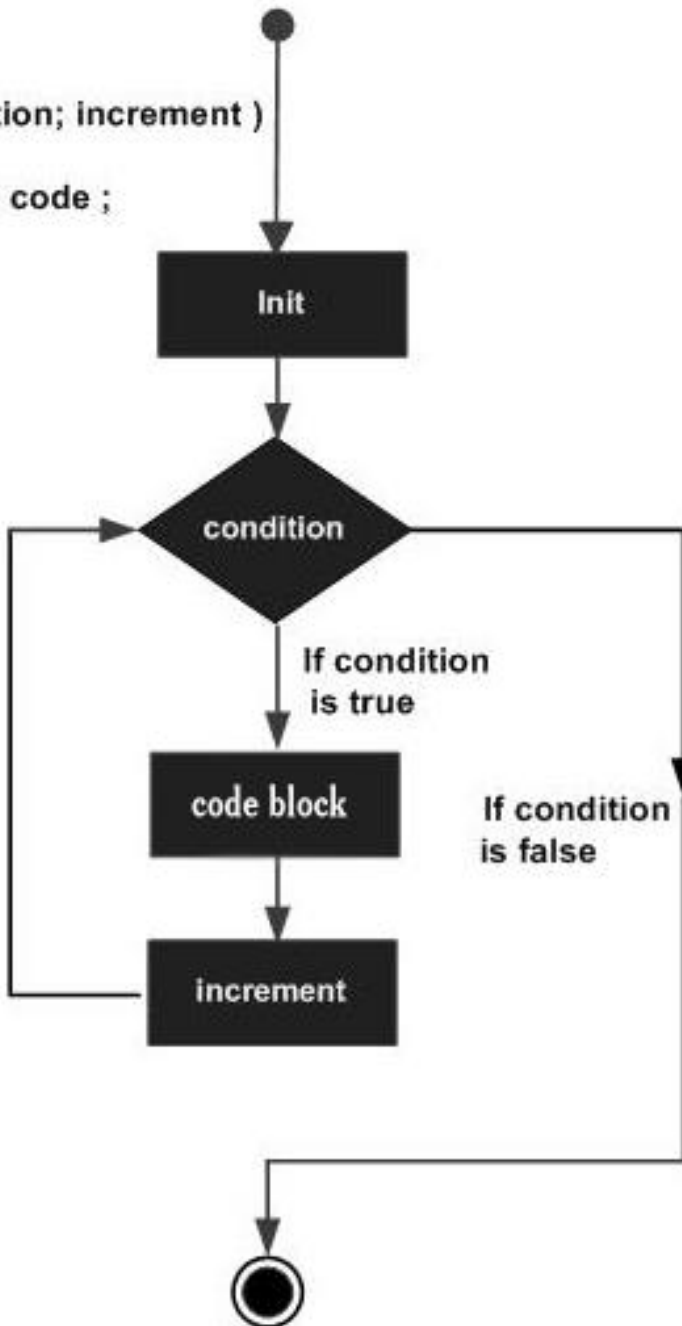
```
for( init; condition; increment )  
{  
    conditional code ;  
}
```



- Thực thi một dãy các lệnh nhiều lần và tóm tắt các đoạn code mà quản lý biến vòng lặp
- Thường được sử dụng như một vòng lặp vô hạn khi một điều kiện không bao giờ false. Nghĩa là **for( ; ; )** (không có điều kiện)  
Lưu ý: vòng lặp này dừng khi bấm tổ hợp phím Ctrl+C

# FOR

```
for( init; condition; increment )
{
    conditional code ;
}
```



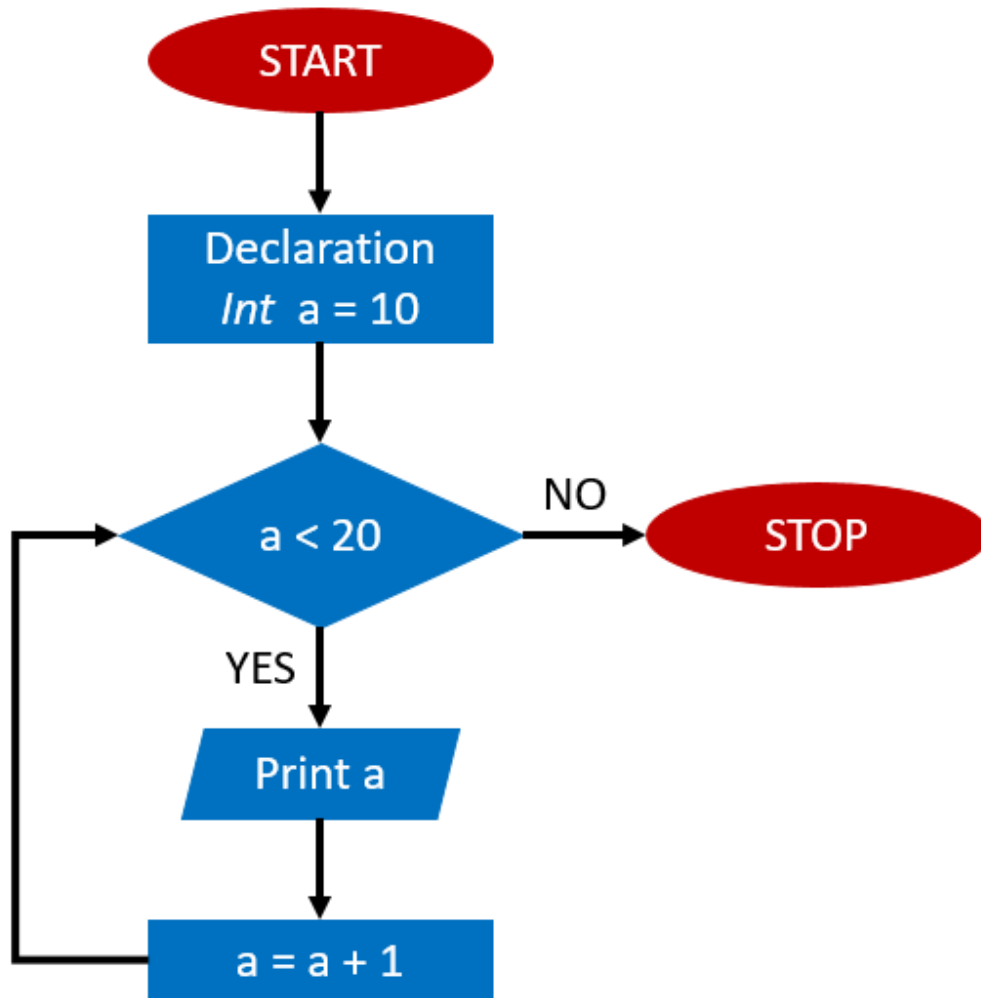
Dạng for:

```
for(khoi_dau;dieu_kien_lap;
buoc_nhay) {
    //dong lenh thuc hien
    khi dieu kien dung
}
```

- Đây là dạng vòng lặp cho phép viết một cách hiệu quả việc thực hiện một số lần lặp cụ thể

# FOR\_ví dụ

14



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

```
int main () {
```

```
    int a;
```

```
    /* for loop execution */
```

```
    for( a = 10; a < 20; a = a + 1 ){  
        printf("value of a: %d\n", a);  
    }
```

```
    return 0;
```

```
}
```

```
value of a: 10  
value of a: 11  
value of a: 12  
value of a: 13  
value of a: 14  
value of a: 15  
value of a: 16  
value of a: 17  
value of a: 18  
value of a: 19
```



# FOR\_ví dụ

15

```
#include <stdio.h>
int main(){

    for (int i = 1; i < 10; i++){
        printf("%d\n",i);
    }

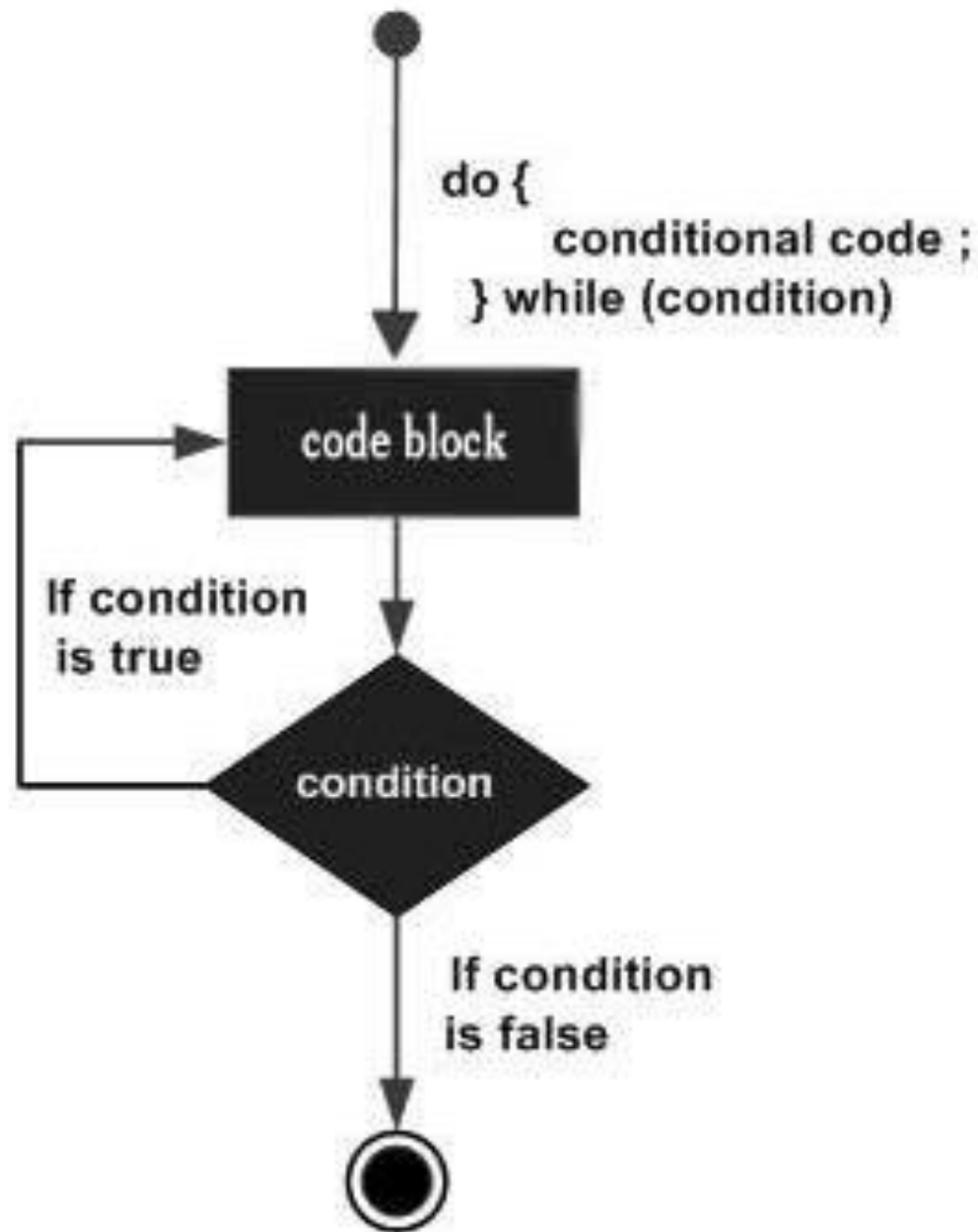
    for (int k = 0; k <= 10; k = k + 2){
        printf("%d\n",k);
    }

    for (int i = 1, j = 2; i + j < 10; i++, j += 2){
        printf("%d + %d = %d\n",i,j,i+j);
    }

    return 0;
}
```

# DO...WHILE

16

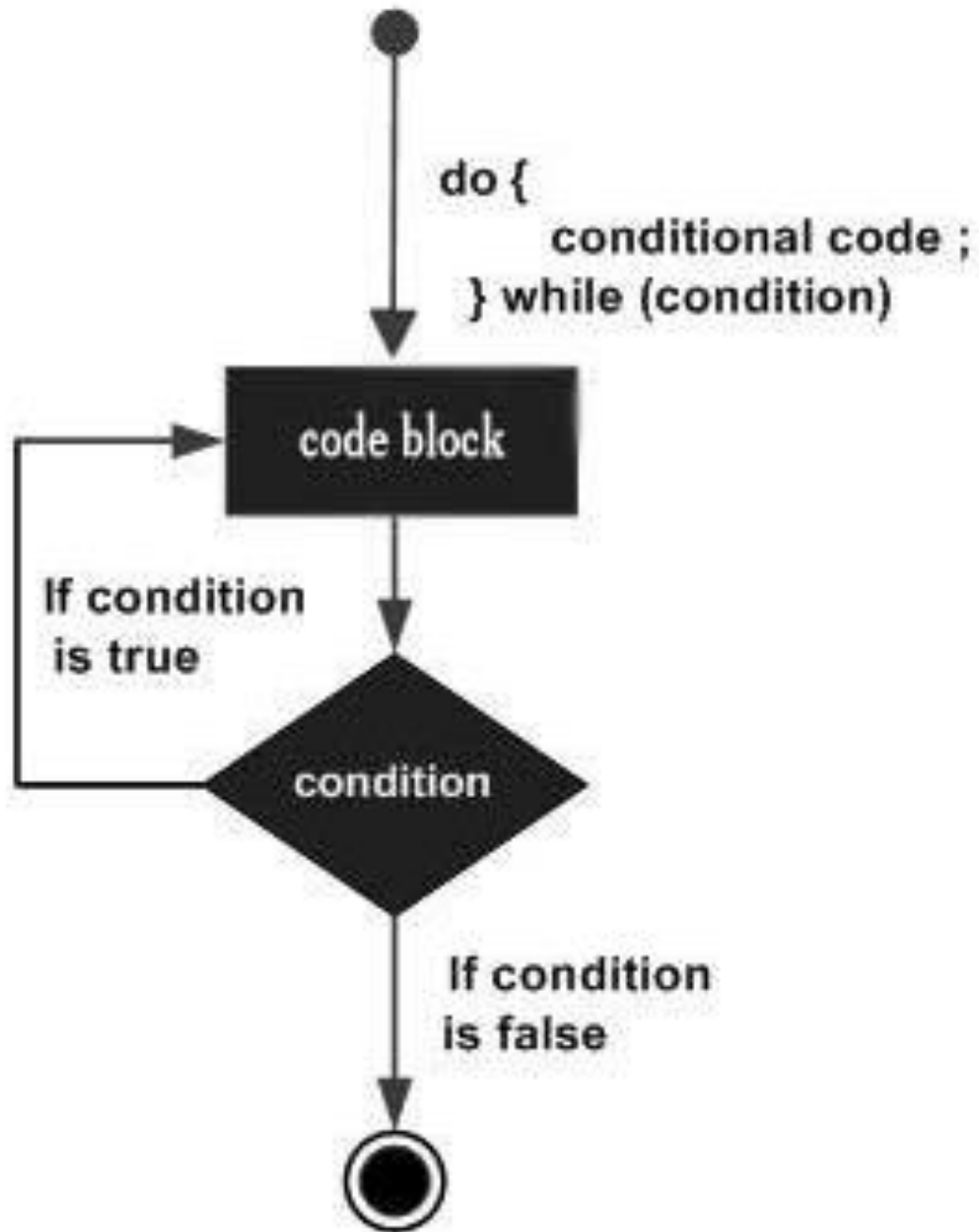


- Giống while nhưng khác là điều kiện được kiểm tra ở cuối thân vòng lặp và sẽ thực hiện thân vòng lặp ít nhất 1 lần
- Lặp lại một hoặc nhóm các lệnh khi điều kiện đã cho là đúng

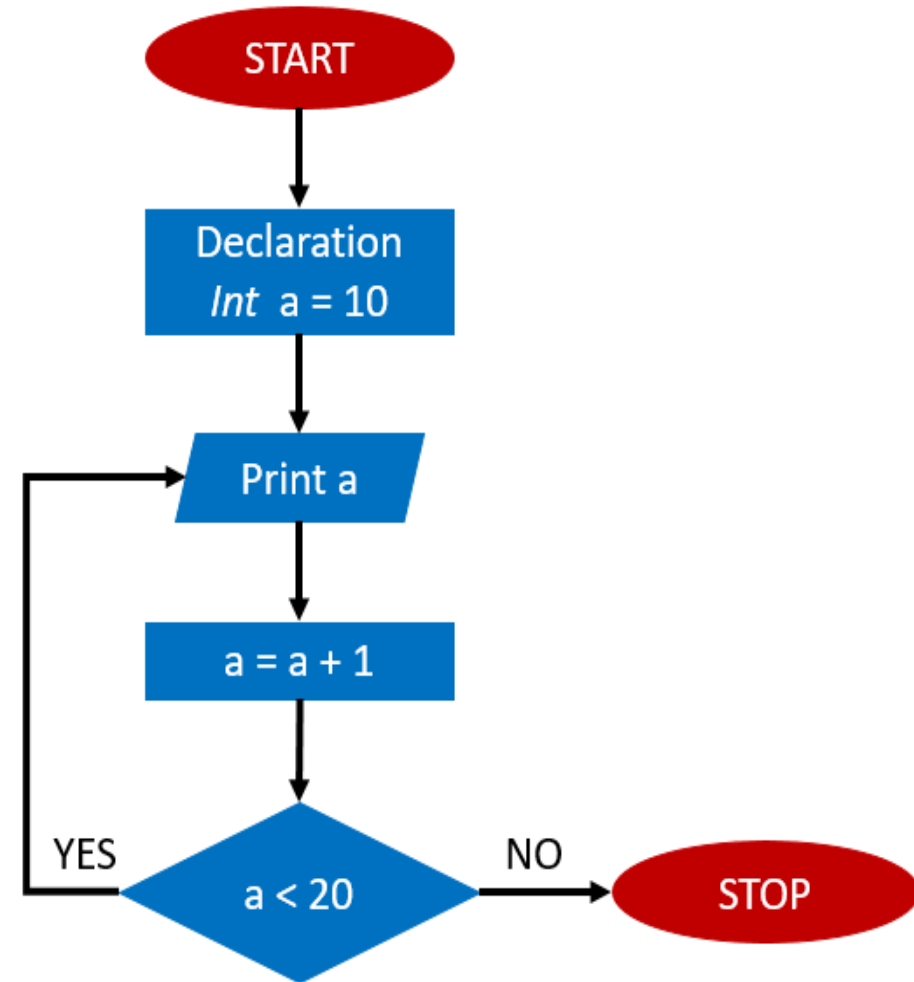
# DO...WHILE

Dạng do...while:

```
do {  
    //dong lenh thuc hien  
}while(dieu_kien)
```



# DO...WHILE\_ví dụ



```

#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* do loop execution */
    do {
        printf("value of a: %d\n", a);
        a = a + 1;
    }while( a < 20 );

    return 0;
}
  
```

```

value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
  
```

# DO...WHILE\_ví dụ

19

```
#include <stdio.h>
int main(){

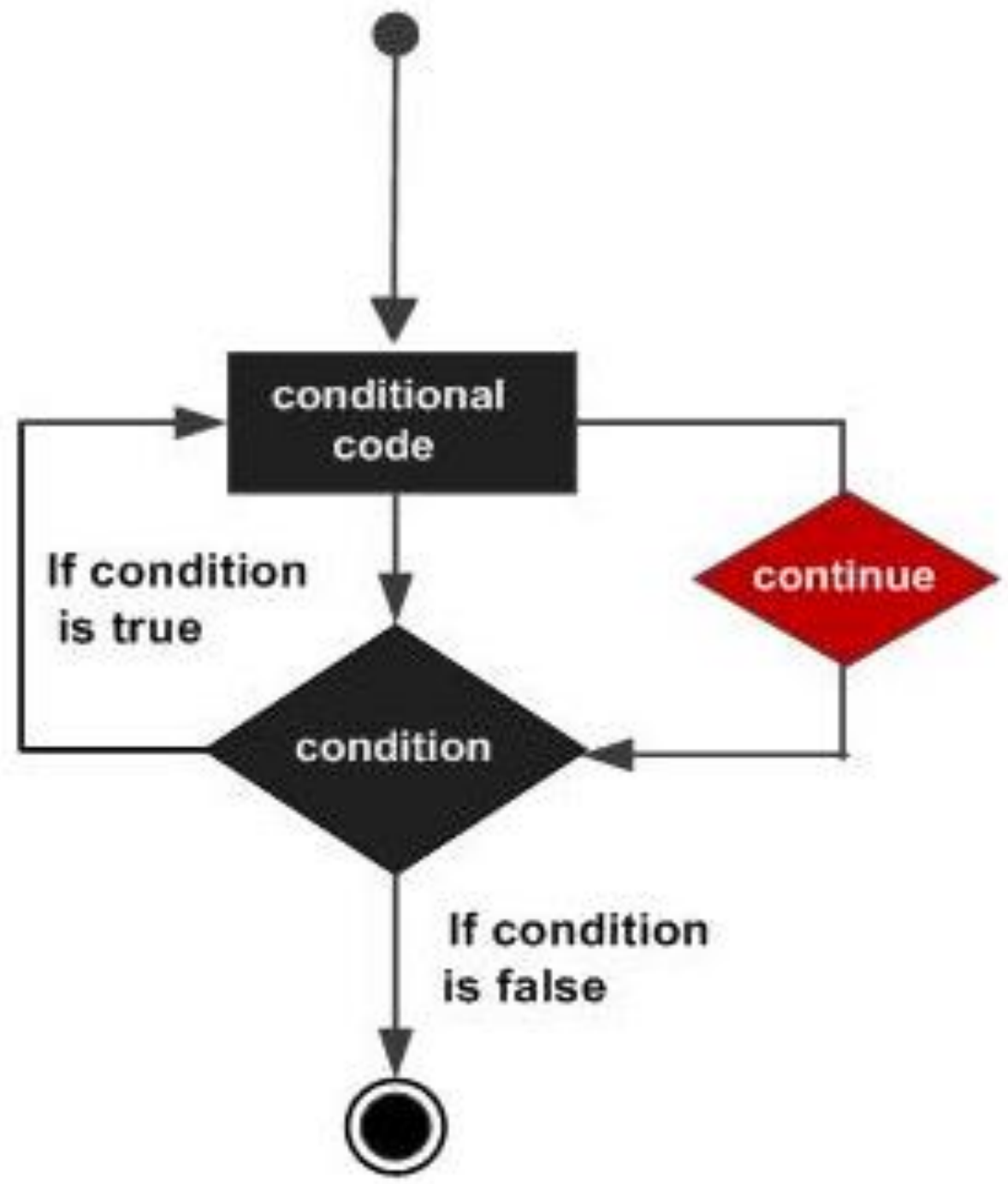
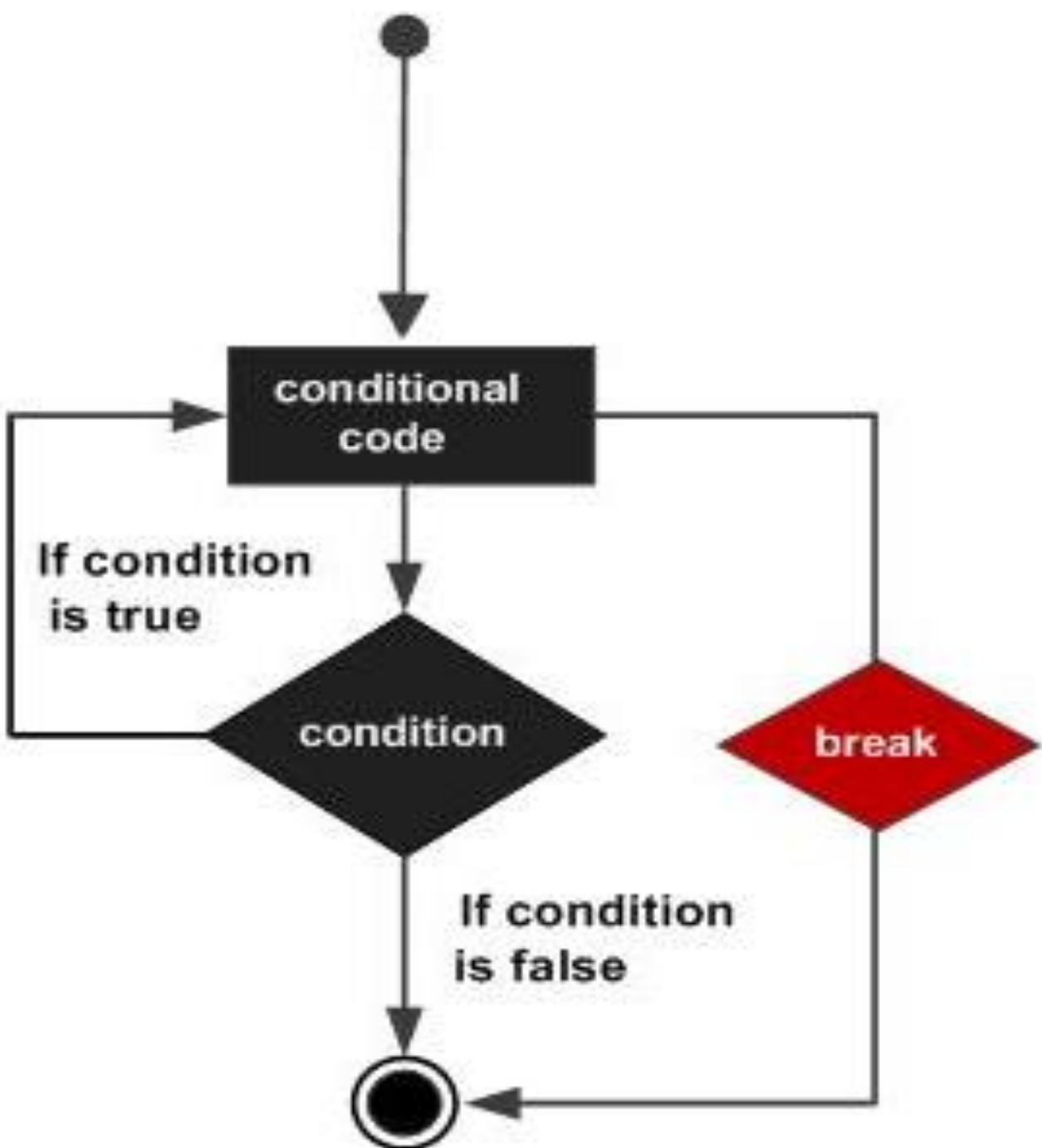
    int n;
    do{
        printf("Nhap n: ");
        scanf("%d",&n);
    }while(n < 5 || n > 10);

    return 0;
}
```

# CÁC LỆNH ĐIỀU KHIỂN VÒNG LẶP

<b>break</b>	Kết thúc <b>vòng lặp ngay lập tức</b> hoặc lệnh <b>switch</b> và chuyển sang thực thi vòng lặp hoặc lệnh switch ngay sau nó
<b>continue</b>	khi gặp lệnh này thì chương trình sẽ bỏ qua các câu lệnh ở dưới nó (trong cùng một câu lệnh lặp) để thực hiện vòng lặp mới
<b>goto</b>	chuyển tới lệnh được gán => lệnh này được khuyên là hạn chế sử dụng trong chương trình do tính phức tạp hơn





# CÁC LỆNH ĐIỀU KHIỂN

## VÒNG LẶP\_ví dụ

22

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* while loop execution */
    while( a < 20 ) {

        printf("value of a: %d\n", a);
        a++;

        if( a > 15) {
            /* terminate the loop using break statement */
            break;
        }

    }

    return 0;
}
```

```
#include <stdio.h>

int main () {

    /* local variable definition */
    int a = 10;

    /* do loop execution */
    do {

        if( a == 15) {
            /* skip the iteration */
            a = a + 1;
            continue;
        }

        printf("value of a: %d\n", a);
        a++;

    } while( a < 20 );

    return 0;
}
```

# BÀI TẬP ÁP DỤNG 3A<sup>23</sup>

1. Sử dụng vòng lặp for và while in các số từ 1-10 theo 2 thứ tự tăng dần và giảm dần
2. Nhập 2 số nguyên a,b. In các số nguyên từ a tới b.  
(ví dụ: a = -2,b=0 thì in ra -2 -1 0.  
Nếu a=4,b=2 thì in ra 4 3 2)
3. Nhập số nguyên n bất kỳ. Hãy cho biết n có phải là số chính phương hay không? (số chính phương là số khi lấy căn bậc 2 có kết quả là nguyên)
4. In một bảng nhân của một số bất kỳ từ 1-10 và hiển thị kết quả. (gợi ý: sử dụng một vòng lặp và tăng dần giá trị của số nhân lên)  
vd kết quả in ra:  $3 \times 1 = 3$

$$3 \times 2 = 6 \dots$$

# BÀI TẬP ÁP DỤNG 3B

24

5. In bảng cửu chương rút gọn là bảng có hàng là kết quả của phép nhân một số với các giá trị từ 1 - 10. Chúng ta sẽ in 9 hàng tương ứng với các số từ 2 đến 10. (gợi ý: vòng lặp bên ngoài điều khiển số hàng và vòng lặp bên trong cho số cột của bảng)

In bang cuu chuong rut gon:

2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

6. In các số chẵn hoặc lẻ trong dãy số từ 1 tới 10.

7. Nhập số nguyên dương n. Hãy in ra các ước số của n và cho biết n có tổng cộng bao nhiêu ước. Từ đó hãy kiểm tra xem n có phải là số nguyên tố hay không.

# BÀI TẬP ÁP DỤNG 3C

25

Các bài tập từ bài 8, hãy dùng vòng lặp để cho phép người dùng nhập lại  $n$  nếu  $n$  không thỏa điều kiện ban đầu:

8. Nhập số nguyên  $n \geq 0$ . Hãy in ra số  $n$  theo thứ tự ngược lại.  
(ví dụ:  $n = 123$  thì in ra số 321. Nếu người dùng nhập số âm thì cho phép nhập lại  $n$ )

9. Nhập số nguyên  $n \geq 0$ . Hãy tính  $n! = 1 * 2 * 3 * \dots * n$  (lưu ý  $0! = 1$ )

10. Nhập số nguyên  $n \geq 0$ . Hãy tính:

$$\begin{aligned} a) S &= \sum_{k=0}^n k! = 0! + 1! + 2! + \dots + n! \\ b) S &= \sum_{k=0}^n \frac{1}{k!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \dots + \frac{1}{n!} \end{aligned}$$



# FUNCTIONS\_HÀM

-



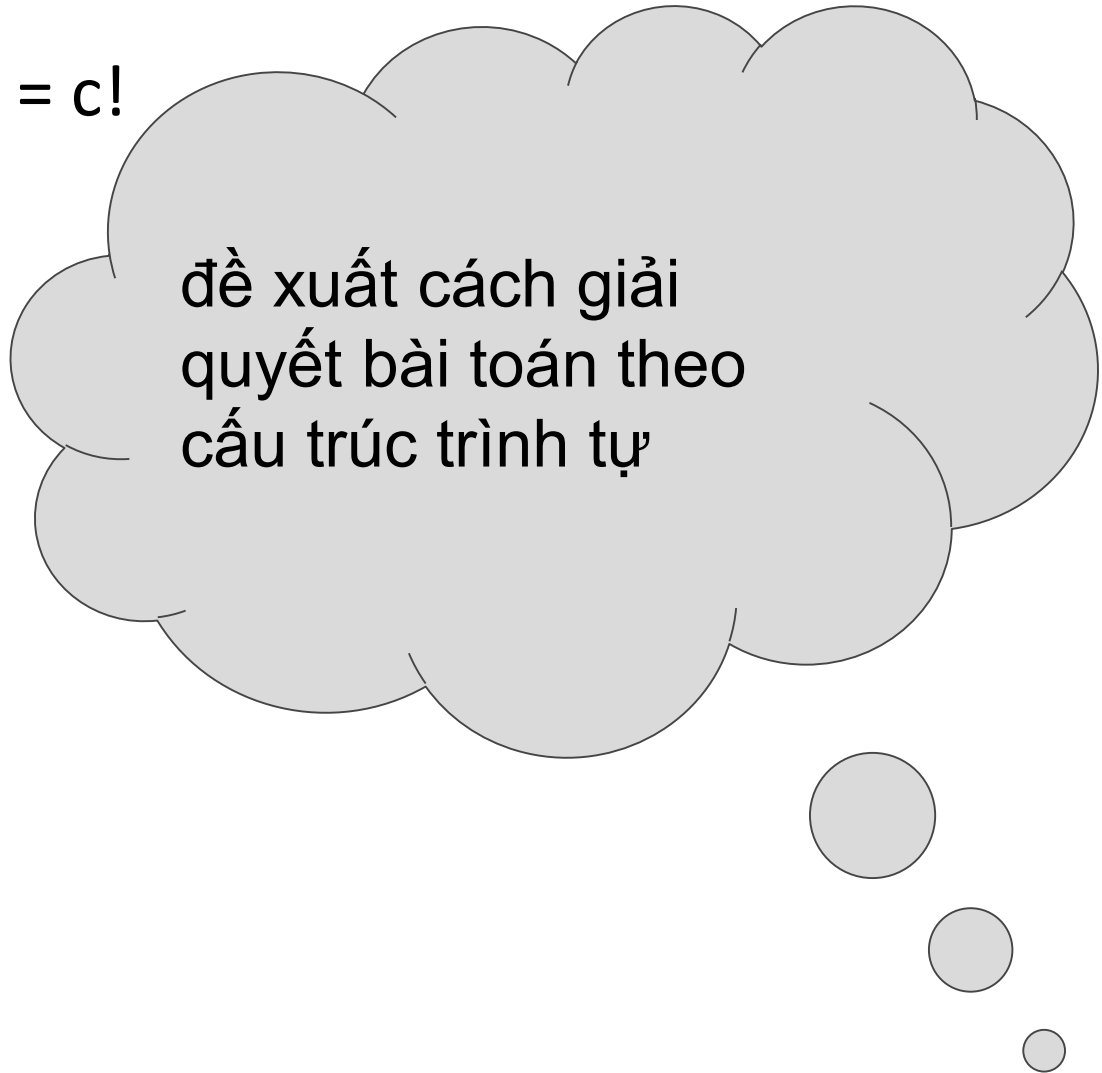
# FUNCTIONS\_bài toán minh họa

- Viết chương trình tính tổng  $S = a! + b! + c!$  với  $a, b$  và  $c$  là 3 số nguyên dương cho trước.

# FUNCTIONS\_bài toán minh họa

3 đoạn lệnh tính tổng  $s1 = a!$ ,  $s2 = b!$ ,  $s3 = c!$

```
// Tính  $s1 = a! = 1*2*...*a$   
int s1 = 1;  
for (int i=1; i<=a; i++) {s1 = s1*i;}  
// Tính  $s2 = b! = 1*2*...*b$   
int s2 = 1;  
for (int i=1; i<=b; i++) {s2 = s2*i;}  
// Tính  $s3 = c! = 1*2*...*c$   
int s3 = 1;  
for (int i=1; i<=c; i++) {s3 = s3*i;}
```

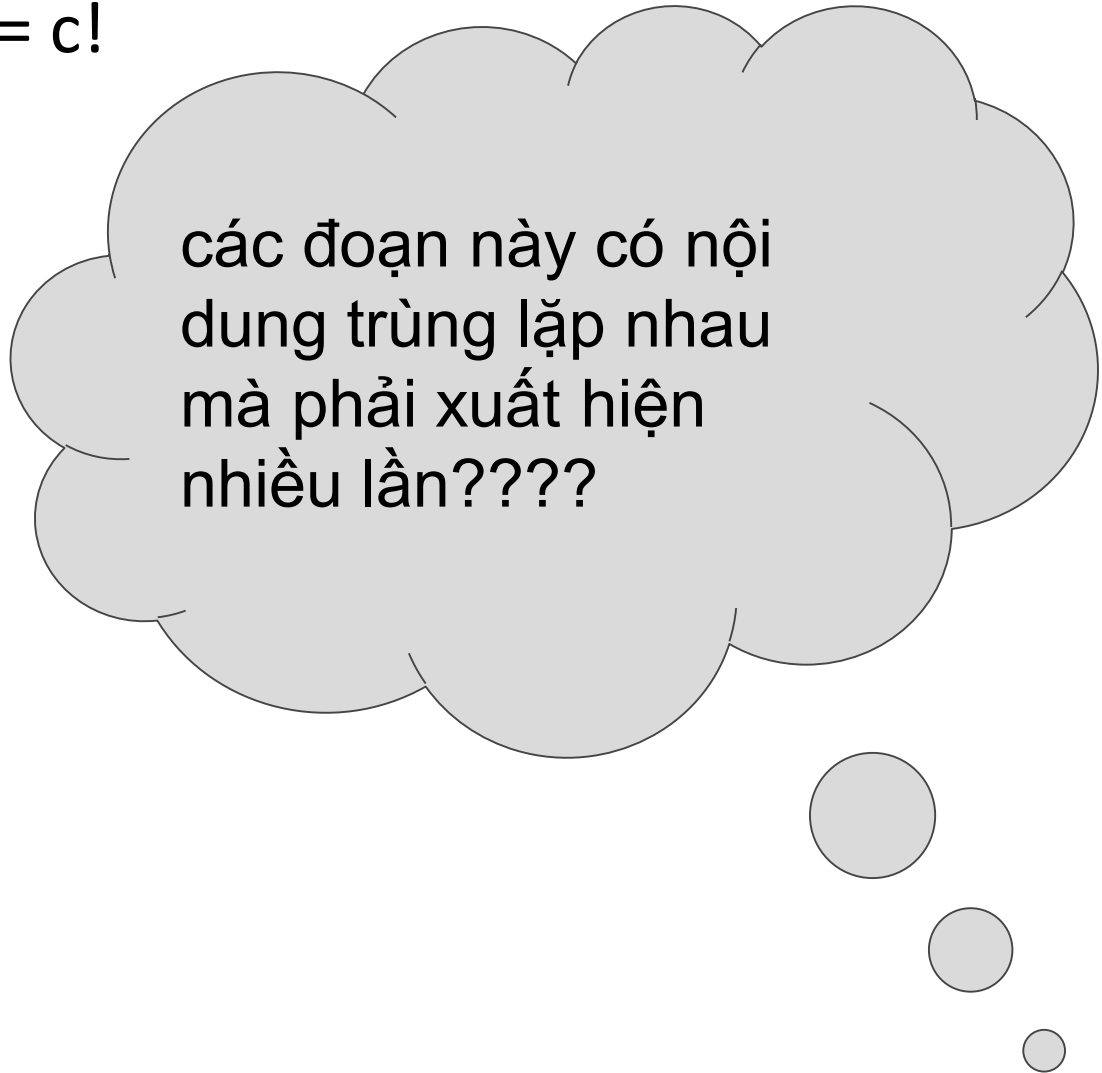


đề xuất cách giải  
quyết bài toán theo  
cấu trúc trình tự

# FUNCTIONS\_bài toán minh họa

3 đoạn lệnh tính tổng  $s1 = a!$ ,  $s2 = b!$ ,  $s3 = c!$

```
// Tính  $s1 = a! = 1*2*...*a$   
int s1 = 1;  
for (int i=1; i<=a; i++) {s1 = s1*i;}  
// Tính  $s2 = b! = 1*2*...*b$   
int s2 = 1;  
for (int i=1; i<=b; i++) {s2 = s2*i;}  
// Tính  $s3 = c! = 1*2*...*c$   
int s3 = 1;  
for (int i=1; i<=c; i++) {s3 = s3*i;}
```



các đoạn này có nội  
dung trùng lặp nhau  
mà phải xuất hiện  
nhiều lần????

# FUNCTIONS\_bài toán minh họa

- 

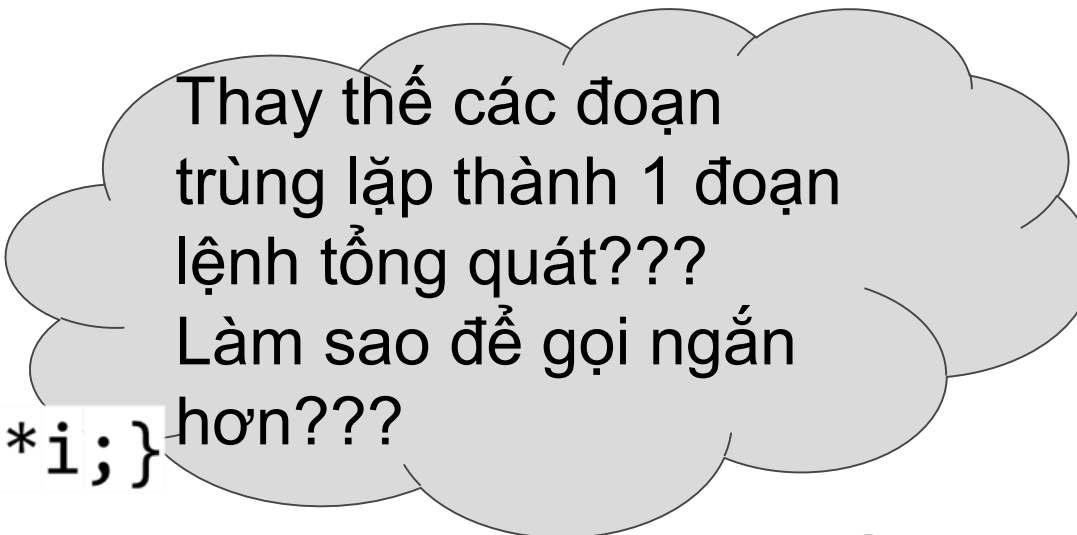
Đoạn lệnh tính giai thừa tổng quát  $s = n! = 1 * 2 * \dots * n$

với  $n = a, b$  hoặc  $c$

```
// Tính  $s = n! = 1*2*\dots*n$ 
```

```
int s = 1;
```

```
for (int i=1; i<=n; i++) {s = s*i;}
```



Thay thế các đoạn  
trùng lặp thành 1 đoạn  
lệnh tổng quát???  
Làm sao để gọi ngắn  
hơn???

# FUNCTIONS\_Khái niệm Hàm

- Là một đoạn chương trình có **tên, đầu vào và đầu ra**
- Có **chức năng giải quyết** một số vấn đề chuyên biệt cho chương trình chính
- có thể **gọi được nhiều lần** với các tham số khác nhau
- được sử dụng khi có nhu cầu:  
tái sử dụng hay sửa lỗi - cải tiến

# FUNCTIONS\_Cú pháp

```
<kiểu trả về> <tên hàm>([<danh sách tham số>])  
{  
    <các câu lệnh>  
    [return <giá trị>;]  
}
```

- <kiểu trả về> : kiểu bất kỳ của C (**char**, **int**, **long**, **float**,...). Nếu không trả về thì là **void**.
- <tên hàm>: theo quy tắc đặt tên định danh.
- <danh sách tham số> : tham số hình thức đầu vào giống khai báo biến, cách nhau bằng dấu phẩy (,)
- <giá trị> : trả về cho hàm qua lệnh **return**



# FUNCTIONS\_các bước viết hàm

- Cần xác định các thông tin:



# FUNCTIONS\_bài toán minh họa

- 

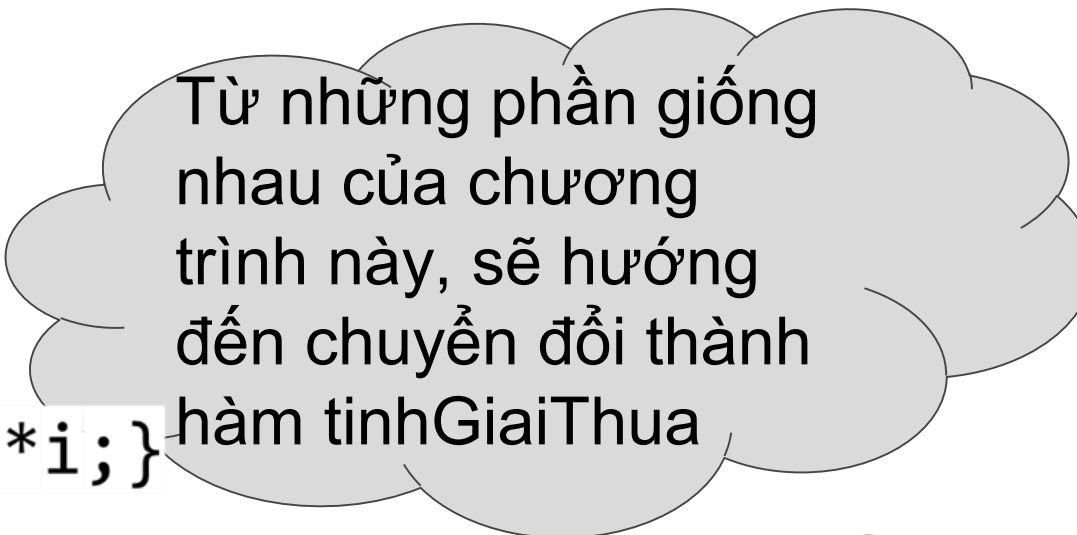
Đoạn lệnh tính giai thừa tổng quát  $s = n! = 1 * 2 * \dots * n$

với  $n = a, b$  hoặc  $c$

```
// Tính  $s = n! = 1 * 2 * \dots * n$ 
```

```
int s = 1;
```

```
for (int i=1; i<=n; i++) {s = s*i;}
```



Từ những phần giống nhau của chương trình này, sẽ hướng đến chuyển đổi thành hàm tínhGiaiThua

# FUNCTIONS\_ví dụ

Viết hàm tính giai thừa số nguyên dương  $n$   
(dựa theo ví dụ)

Tên hàm: ***tingGiaiThua***

Chức năng: ***tính*** và ***trả về*** giai thừa của số nguyên dương  $n$

Input: ***một*** số nguyên dương  $n$

Output: ***một*** số nguyên là giai thừa của  $n$

# FUNCTIONS\_ví dụ

Viết hàm tính giai thừa số nguyên dương n  
(dựa theo ví dụ)

```
//Ham tinh giai thua so nguyen duong n  
//Input: so nguyen duong n  
//Output: giai thua s cua n  
int tinhGiaiThua(int n){  
    int s = 1;  
    for(int i=1; i<=n ; i++){  
        s = s*i;  
    }  
    return s;  
}
```

# FUNCTIONS\_cách gọi hàm

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

```
int tinhGiaiThua(int n);
```

```
int main()
```

```
{
```

```
    int n = 5;
```

```
    int s;
```

```
    s = tinhGiaiThua(n);
```

```
    printf("%d! = %d\n", n, s);
```

```
    return 0;
```

```
}
```

```
//Ham tinh giai thua so nguyen duong n
```

```
//Input: so nguyen duong n
```

```
//Output: giai thua s cua n
```

```
int tinhGiaiThua(int n){
```

```
    int s = 1;
```

```
    for(int i=1; i<=n ; i++){
```

```
        s = s*i;
```

```
    }
```

```
    return s;
```

```
}
```

```
int tinhGiaiThua(int n)
```

```
s = tinhGiaiThua( n );
```

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

```
int tinhGiaiThua(int n);
```

```
int main()
```

```
{  
    int n = 5;  
    int s;  
    s = tinhGiaiThua(n);  
    printf("%d! = %d\n", n, s);  
  
    return 0;  
}
```

*//Ham tinh giai thua so nguyen duong n*

*//Input: so nguyen duong n*

*//Output: giai thua s cua n*

```
int tinhGiaiThua(int n){  
    int s = 1;  
    for(int i=1; i<=n ; i++){  
        s = s*i;  
    }  
    return s;  
}
```

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

*//Ham tinh giai thua so nguyen duong n*

*//Input: so nguyen duong n*

*//Output: giai thua s cua n*

```
int tinhGiaiThua(int n){  
    int s = 1;  
    for(int i=1; i<=n ; i++){  
        s = s*i;  
    }  
    return s;  
}
```

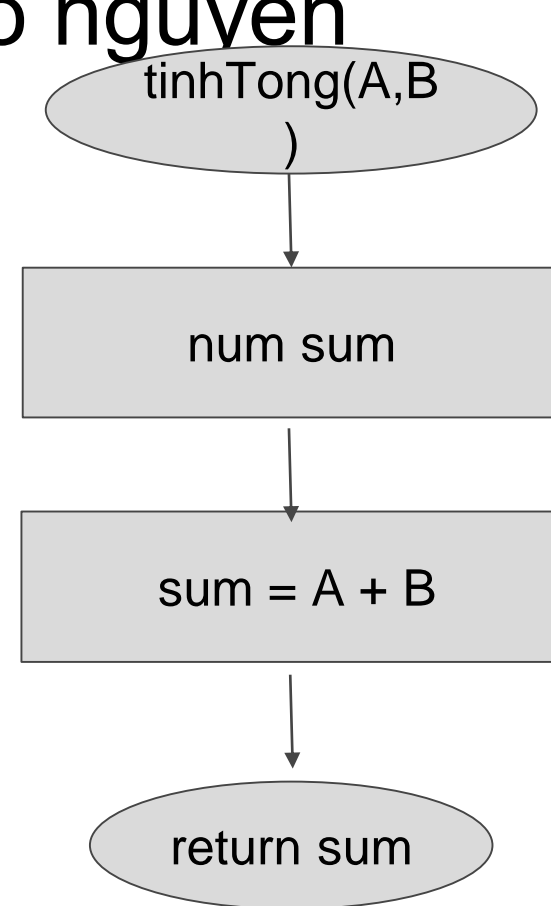
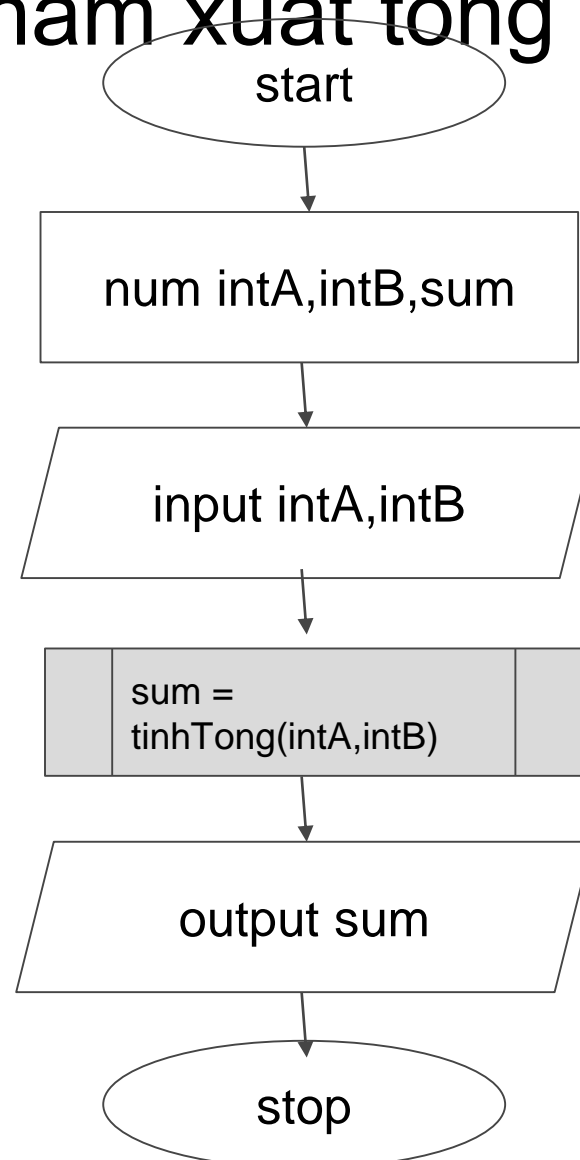
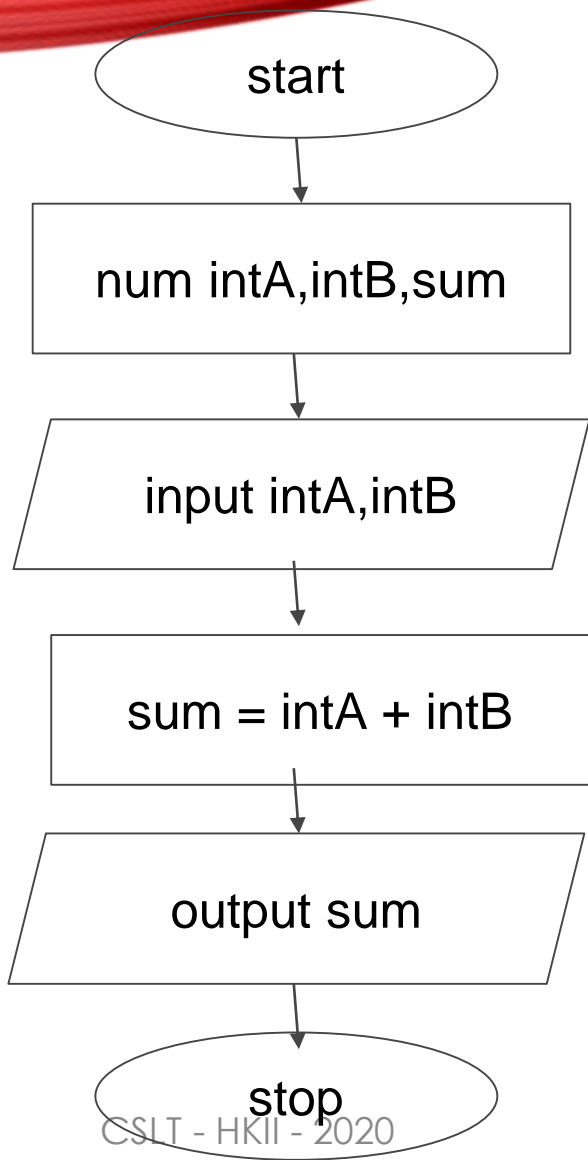
```
void main()
```

```
{  
    int n = 5;  
    int s;  
    s = tinhGiaiThua(n);  
    printf("%d! = %d\n", n, s);  
}
```



# FUNCTIONS\_ví dụ 2 <sup>39</sup>

## Viết hàm xuất tổng 2 số nguyên



# FUNCTIONS\_ví dụ 2 <sup>40</sup>

## Viết hàm xuất tổng 2 số nguyên

Tên hàm:  ***tínhTong***

Chức năng:  ***tính*** và  ***trả về*** tổng của 2 số nguyên dương a,b

Input: số nguyên dương a,b

Output:  ***một*** số nguyên là sum của a và b

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

//Ham tinh tong 2 so nguyen a ,b
//Input: 2 so nguyen a,b
//Output: sum cua a,b
int tinhTong(int a,int b){
    int sum;        //khai báo biến
    sum = a+b;      //xử lý
    return sum;     //trả về giá trị
}

void main(){
    int a,b,sum;
    printf("Nhap a,b: ");
    scanf("%d %d",&a,&b);
    sum = tinhTong(a,b);
    printf(" tong cua 2 so nguyen %d va %d la %d",a,b,sum);
}
```

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

```
//khai báo hàm
```

```
int timGTLN(int so1,int so2);
```

```
//chương trình chính
```

```
int main()
```

```
{
```

```
    //định nghĩa và khai báo biến cục bộ
```

```
    int a = 667;
```

```
    int b = 7028;
```

```
    int ketqua;
```

```
    // gọi hàm
```

```
    ketqua = timGTLN(a,b);
```

```
    printf("Gia tri lon nhat trong 2 gia tri la: %d\n",ketqua);
```

```
    return 0;
```

```
}
```

```
//định nghĩa hàm
```

```
//Hàm tìm giá trị lớn nhất giữa 2 số a,b
```

```
//Input: 2 số nguyên a,b
```

```
//Output: giá trị lớn nhất giữa a,b
```

```
int timGTLN(int so1, int so2 ){
```

```
    //khai báo biến cục bộ
```

```
    int ketqua;
```

```
    // nội dung xử lý
```

```
    if(so1 > so2)
```

```
        ketqua = so1;
```

```
    else
```

```
        ketqua = so2;
```

```
    return ketqua;
```

# FUNCTIONS\_ví dụ 3

## Tìm số lớn nhất giữa 2 số

### nguyên

**Tên hàm:** *timGTLN*

**Chức năng:** tìm giá trị lớn nhất giữa 2 số nguyên a,b

**Input:** số nguyên dương a,b

**Output:** giá trị lớn nhất giữa 2 số nguyên a,b

# BIẾN TOÀN CỤC, BIẾN CỤC BỘ và THAM SỞ CHÍNH THỨC

# Khái niệm

<b>Biến cục bộ</b>	Bên trong một hàm hoặc một khối code	<ul style="list-style-type: none"> <li>• Được sử dụng bởi các lệnh <b>trong hàm</b> hoặc khối code</li> <li>• <b>Không</b> được sử dụng bên ngoài của hàm</li> <li>• Nếu một chương trình có biến toàn cục và cục bộ <b>trùng tên thì biến cục bộ trong hàm được ưu tiên sử dụng.</b></li> </ul>
<b>Biến toàn cục</b>	Bên ngoài tất cả các hàm	<ul style="list-style-type: none"> <li>• Thường được định nghĩa <b>ở ngoài các hàm và ở phần đầu chương trình</b></li> <li>• Chứa các giá trị trong thời gian chương trình chạy</li> <li>• <b>Được truy cập bởi bất kỳ hàm nào</b> được định nghĩa trong chương trình</li> </ul>
<b>Tham số chính thức</b>	Trong định nghĩa các hàm	<ul style="list-style-type: none"> <li>• Thường nói thông dụng thì đây là biến <b>local</b> trong hàm</li> <li>• Giá trị này <b>trong hàm</b> sẽ được <b>ưu tiên hơn biến toàn cục</b></li> </ul>

# Ví dụ 1

```
#include <stdio.h>

//Khai báo biến toàn cục
int g;

void main(){
    //Khai báo biến cục bộ
    int a,b;

    //Khởi tạo giá trị biến
    a = 10;
    b = 20;
    g = a+b;

    printf("Gia tri cua:\n a = %d\n b = %d\n g = %d\n",a,b,g);
}
```

```
$gcc -o main *.c
```

```
$main
```

Gia tri cua:

a = 10

b = 20

g = 30



## Ví dụ 2

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

```
//Khai báo biến toàn cục
```

```
int g = 20;
```

```
void main(){
```

```
//Khai báo biến cục bộ
```

```
int g = 10;
```

```
printf("Gia tri cua g = %d\n",g);
```

```
//trong hàm sẽ ưu tiên giá trị cục bộ hơn toàn cục
```

```
}
```

```
$gcc -o main *.c
```

```
$main
```

```
Gia tri cua g = 10
```

# Ví dụ 3

```
#include <stdio.h>

//Khai báo biến toàn cục
int a = 20;

//Hàm tính tổng 2 số nguyên a,b
//Input: 2 số nguyên a,b
//Output: tổng của 2 số a,b
int tinhTong(int a, int b){
    // 2 giá trị a và b lúc này là tham số chính thức của hàm
    tinhTong
    printf("Gia tri a trong ham tinhTong la %d",a);
    printf("Gia tri b trong ham tinhTong la %d",b);
    //Khi gọi hàm 2 giá trị này sẽ được đưa vào từ hàm main
    return a+b;
}

void main(){
    //Khai báo biến cục bộ
    int a = 15;
    int b = 25;
    int c = 0;

    printf("Gia tri cua c truoc khi goi ham = %d\n",c);

    printf("Gia tri cua a trong ham main la %d\n",a);
    //trong hàm sẽ ưu tiên giá trị cục bộ hơn toàn cục

    c = tinhTong(a,b);
    printf("Gia tri cua c sau khi goi ham = %d\n",c);
}
```

Gia tri cua c truoc khi goi ham = 0  
 Gia tri cua a trong ham main la 15  
 Gia tri a trong ham tinhTong la 15  
 Gia tri b trong ham tinhTong la 25  
 Gia tri cua c sau khi goi ham = 40

# Giá trị khởi tạo biến toàn cục và biến cục bộ

Kiểu dữ liệu	Giá trị khởi tạo mặc định
int	0
char	'\0'
float	0
double	0
pointer	NULL

# THE END

