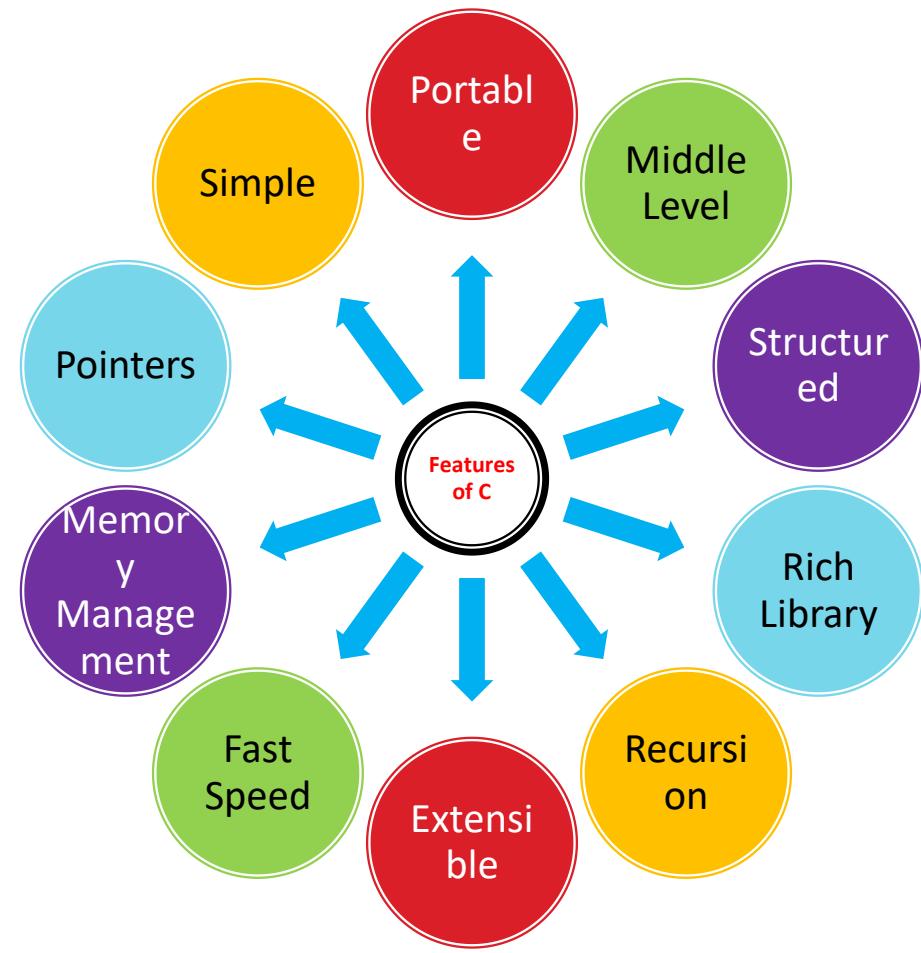


PHƯƠNG PHÁP LẬP TRÌNH



NỘI DUNG MÔN HỌC

1. Tổng quan về giải thuật
2. Ngôn ngữ lập trình C
3. Cấu trúc điều khiển (*Control structures and statements*)
4. Hàm (*Function*)
5. Mảng 1 chiều (*One array dimension*)
6. Chuỗi ký tự (*String*)
7. Mảng hai chiều (*Two array dimension*)
8. Kiểu dữ liệu cấu trúc (*Struct data type*)
9. Kiểu dữ liệu con trỏ (*Pointer data type*)
10. Tập tin (*File*)

Chương 3

CẤU TRÚC ĐIỀU KHIỂN

(Control structures and statements)



MỤC TIÊU CỦA CHƯƠNG

Sau khi hoàn tất chường này, sinh viên có khả năng Hiểu và vận dụng được các cấu trúc điều khiển (rẽ nhánh và cấu trúc lặp) để viết được chương trình trên máy tính.

NỘI DUNG CHƯƠNG

1. Câu lệnh (*statement*) và khối lệnh (*block*)

2. Cấu trúc tuần tự (*Sequence structure*)

Lệnh 1

Lệnh 2

Lệnh 3

...

2. Cấu trúc rẽ nhánh (*Selection structure with one or two branches*)

- if
- if ... else
- if ... else lồng nhau nhiều cấp

3. Cấu trúc lựa chọn (*Selection structure with many branches*) switch

4. Cấu trúc lặp (*Loop structure*)

- for
- while
- do ... while

1. CÂU LỆNH VÀ KHỐI LỆNH TRONG LẬP TRÌNH

1.1. Câu lệnh (*statement*)

- Một câu lệnh xác định một công việc mà chương trình phải thực hiện để xử lý dữ liệu đã được mô tả và khai báo.
- Các câu lệnh được ngăn cách với nhau bởi dấu chấm phẩy (;).
- VD:

```
int n;  
printf("Nhap vao so nguyen n = ");  
scanf("%d",&n);  
printf("Ban vua nhap so %d",n);
```

1. Câu lệnh và khối lệnh trong lập trình

1.2. Khối lệnh

- Một dãy các câu lệnh được bao bởi các dấu { } gọi là một khối lệnh.
- Ví dụ:

```
void main()
{ int n = 10, tong=0;
  for (int i = 1; i < n; i++)
  {
    if (i % 2 == 0)
    {
      tong = tong+i;
    }
  }
  printf("Tong cac so le tu 1 den %d la %d", n, tong);
  getch();
}
```

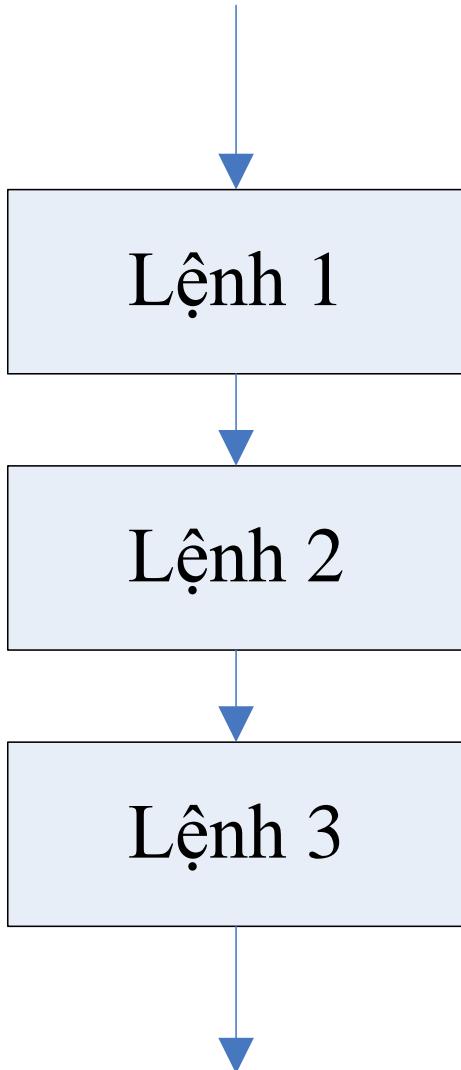
1. Câu lệnh và khối lệnh trong lập trình

1.3. Phạm vi hoạt động của biến trong các khối lệnh

- Tất cả các biến sử dụng trong chương trình đều phải được khai báo trước.
- Phạm vi hoạt động của 1 biến chính là khối lệnh mà biến được khai báo:
 - *Khai báo biến trong hàm*: tầm hoạt động sẽ là toàn bộ hàm đó.
 - *Khai báo biến trong vòng lặp*: thì tầm hoạt động sẽ chỉ là vòng lặp đó.
- Ví dụ:

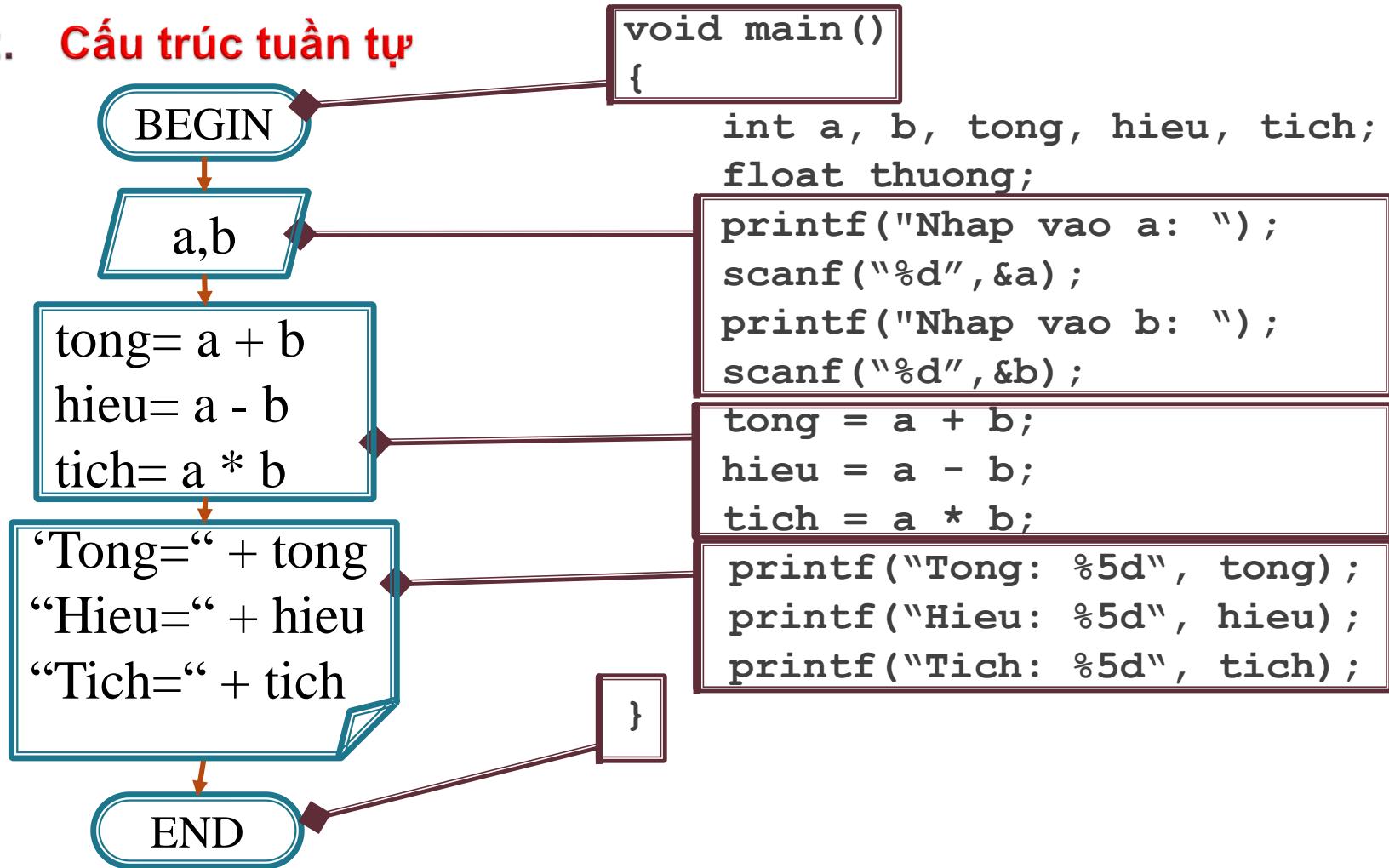
```
int tong=0;
void main()
{
    int n = 10;
    for (int i = 1; i < n; i++)
    {
        if (i % 2 == 0)
        {
            tong = tong+i;
        }
    }
    printf("Ket thuc vong lap for, gia tri cua i=%d", i)
        //Error C2065 'i': undeclared identifier
    printf("Tong cac so le tu 1 den %d la %d",n,tong);
}
```

2. CẤU TRÚC TUẦN TỰ (Sequence structure)



- Là cách tổ chức các lệnh thành từng khối
- Tuần tự thực thi tiến trình, mỗi lệnh được thực thi theo một chuỗi từ trên xuống, xong lệnh này rồi chuyển xuống lệnh kế tiếp.

2. Cấu trúc tuần tự



Bài tập:

- 1.- Cho nhập bán kính.In ra chu vi, diện tích của hình tròn
- 2.- Giả sử có các loại tờ tiền 1đ, 2đ, 5đ, 10đ và 20đ. Viết chương trình cho người dùng nhập vào số tiền, tìm số lượng mỗi loại tiền là bao nhiêu để có số tờ giấy bạc là ít nhất.

3. CẤU TRÚC RẼ NHÁNH

(Selection structure with one or two branches)

3.1. Công dụng: Cấu trúc rẽ nhánh chỉ cho máy tính chọn thực hiện một dãy lệnh nào đó dựa vào kết quả của một điều kiện (biểu thức quan hệ hay biểu thức so sánh)

3.2. Phân loại: Gồm 2 dạng chính:

3.2.1. Chỉ xét trường hợp đúng điều kiện

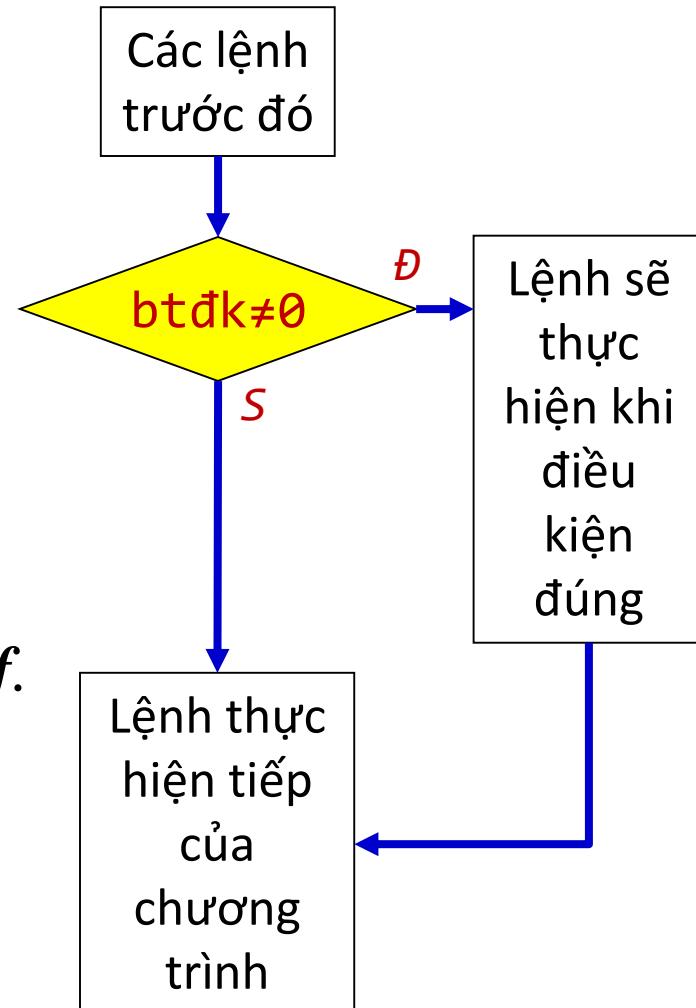
if (biểu thức điều kiện)

{

<khối lệnh>;

}

- Nếu biểu thức điều kiện cho kết quả là **true** thì thực hiện khối lệnh bên trong **if**.
- Khi khối lệnh có nhiều hơn 1 lệnh, bắt buộc khối lệnh này phải được đặt trong cặp dấu ngoặc nhọn { }

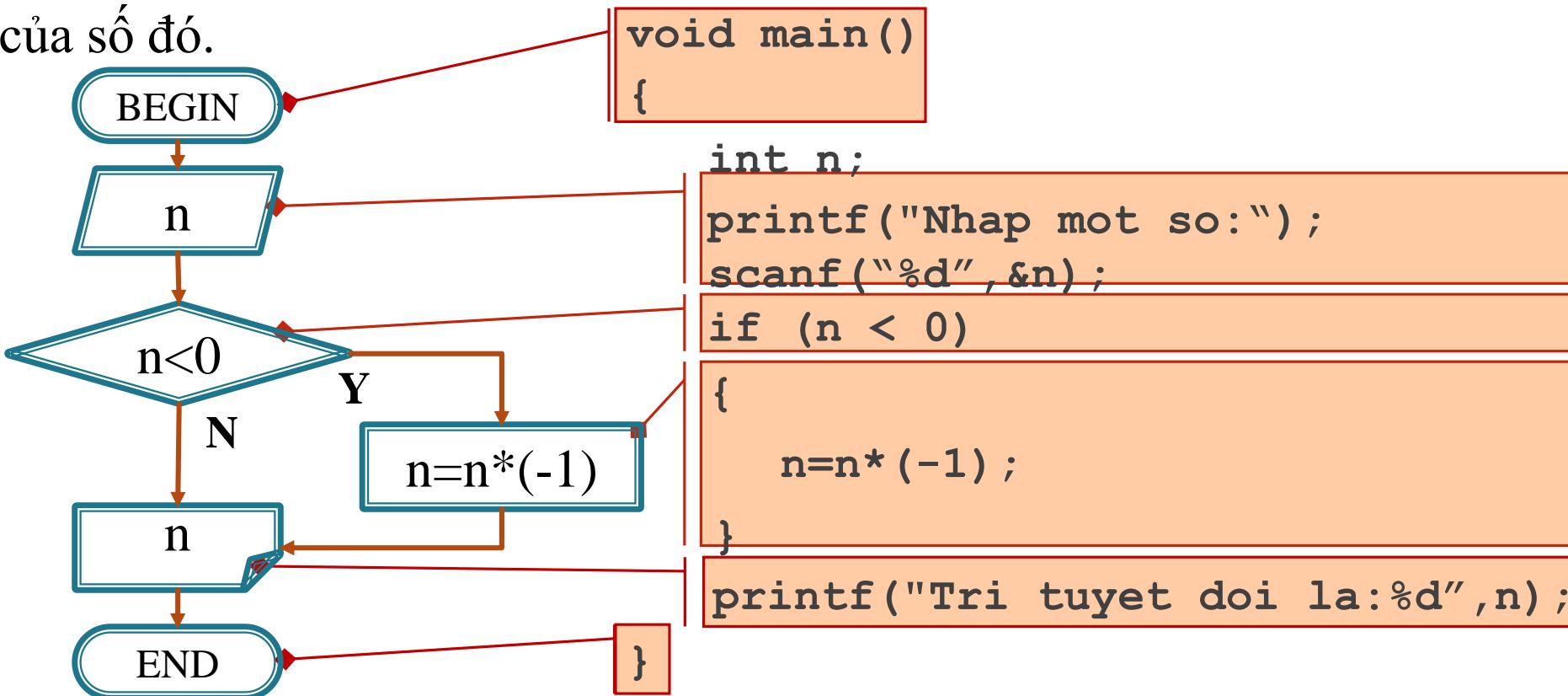


3. Cấu trúc rẽ nhánh

3.2. Phân loại: Gồm 2 dạng chính:

3.2.1. Chỉ xét trường hợp đúng điều kiện

Minh họa: Viết chương trình nhập vào một số, in ra trị tuyệt đối của số đó.



3. Cấu trúc rẽ nhánh

3.2. Phân loại: Gồm 2 dạng chính:

3.2.1. Chỉ xét trường hợp đúng điều kiện

/* Yêu cầu: Nhập 2 số nguyên. Cho biết sự tương quan giữa chúng. */

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

```
void main()
```

```
{ int a, b; // hai biến số nguyên
```

```
printf("Nhập số nguyên thứ nhất: "); scanf("%d", &a);  
printf("Nhập số nguyên thứ hai : "); scanf("%d", &b);
```

if (a != b)	printf("%d không bằng %d\n", a, b);
if (a == b)	printf("%d bằng %d\n", a, b);
if (a < b)	printf("%d nhỏ thua %d\n", a, b);
if (a <= b)	printf("%d nhỏ thua hay bằng %d\n", a, b);
if (a > b)	printf("%d lớn hơn %d\n", a, b);
if (a >= b)	printf("%d lớn hơn hay bằng %d\n", a, b);

```
}
```

Nhập số nguyên thứ nhất: 9
Nhập số nguyên thứ hai : 7

9	không bằng	7
9	lớn hơn	7
9	lớn hơn hay bằng	7

Nhập số nguyên thứ nhất: 9
Nhập số nguyên thứ hai : 9

9	bằng	9
9	nhỏ thua hay bằng	9
9	lớn hơn hay bằng	9

3. Cấu trúc rẽ nhánh

3.2. Phân loại: Gồm 2 dạng chính:

3.2.2. Dạng 2: Xét cả hai trường hợp đúng và sai

if (biểu thức điều kiện)

{

<khởi lệnh 1>;

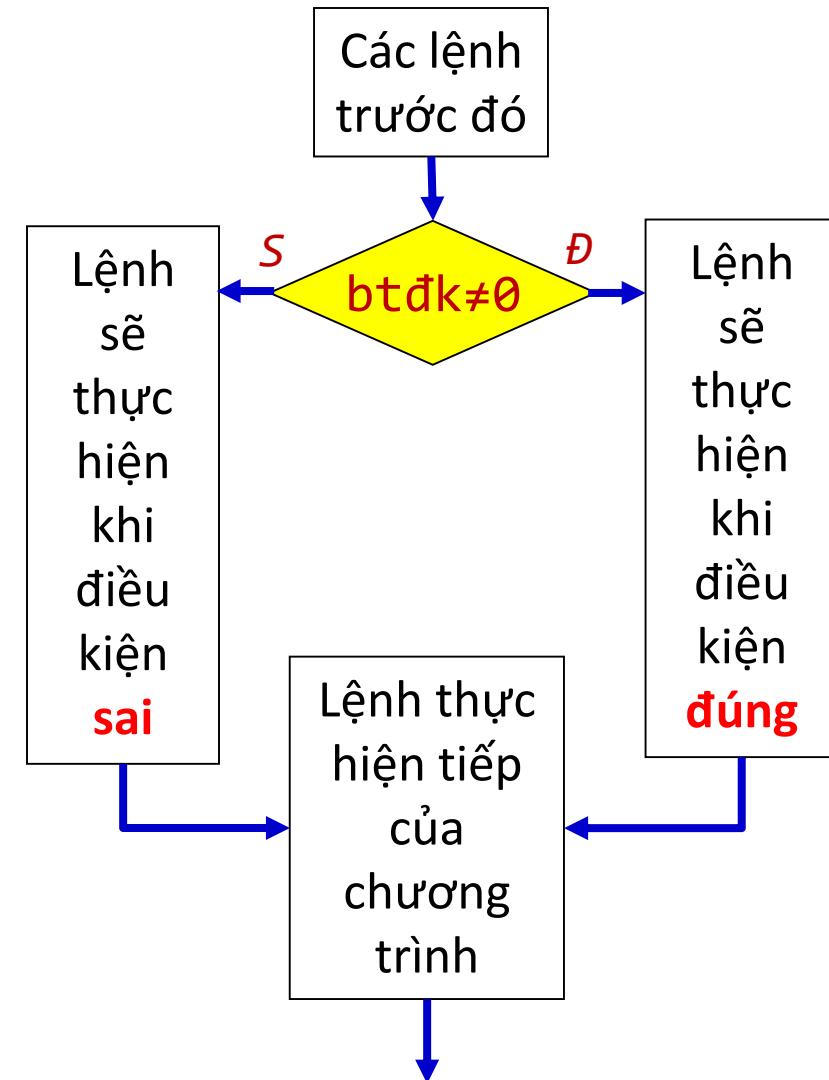
}

else

{

<khởi lệnh 2>;

}

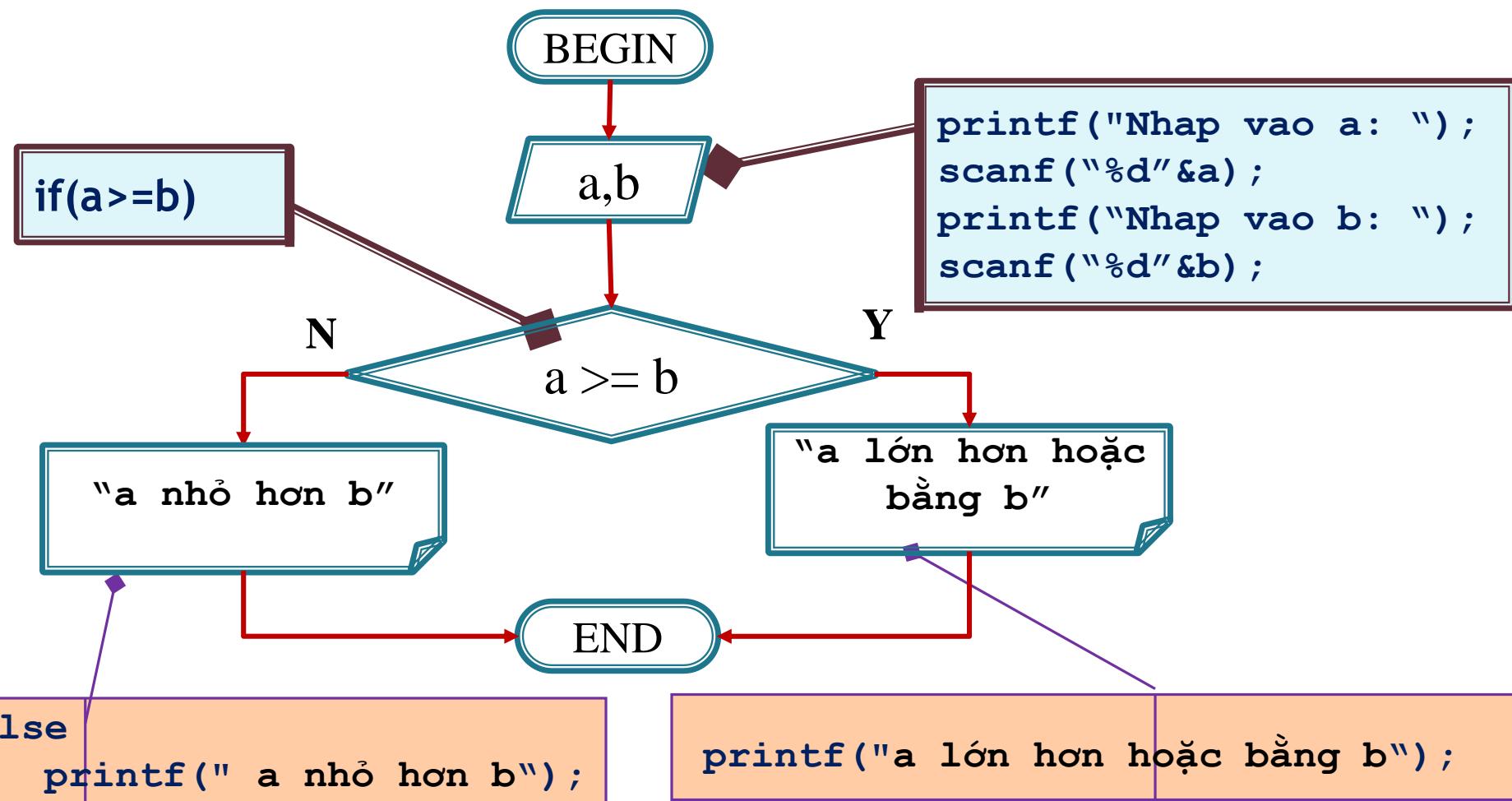


3. Cấu trúc rẽ nhánh

3.2. Phân loại: Gồm 2 dạng chính:

3.2.2. Dạng 2: Xét cả hai trường hợp đúng và sai

Ví dụ 1: Nhập vào số nguyên a và b, cho biết “a là lớn hơn hay bằng b” hay “a nhỏ hơn b”



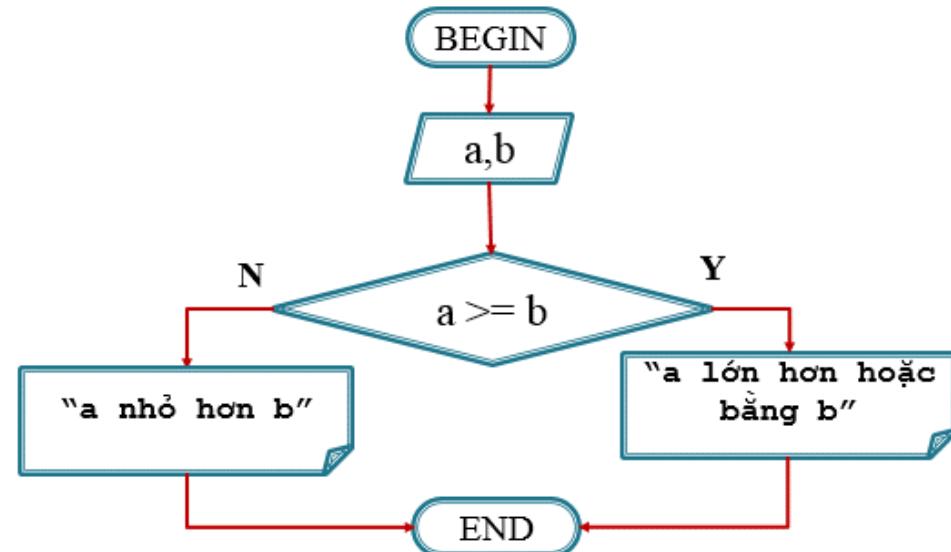
3. Cấu trúc rẽ nhánh

3.2. Phân loại: Gồm 2 dạng chính:

3.2.2. Dạng 2: Xét cả hai trường hợp đúng và sai

Cài đặt Ví dụ 2

```
void main()
{
    int a, b;
    printf("Nhập vào a: ");
    scanf("%d", &a);
    printf("Nhập vào b: ");
    scanf("%d", &b);
    if(a>=b)
    {
        printf(" a lớn hơn hoặc bằng b");
    }
    else
    {
        printf("a nhỏ hơn b");
    }
}
```



Bài tập: Cho nhập 2 số a và b. In ra tổng, hiệu, tích, thương của a và b

3. Cấu trúc rẽ nhánh

3.3. Nhiều lệnh if lồng nhau

else kết nối với lệnh if gần nhất

```
int i = 100;
if(i > 0)
    if(i > 1000)
        printf("i qua lon\n");
    else
        printf("i chap nhan duoc\n");
```

```
int i = -20;
if(i > 0)
{
    if(i > 1000)
        printf("i qua lon\n");
}
else
    printf("i la so am\n");
```

Kết quả:

i chap nhan duoc

Kết quả:

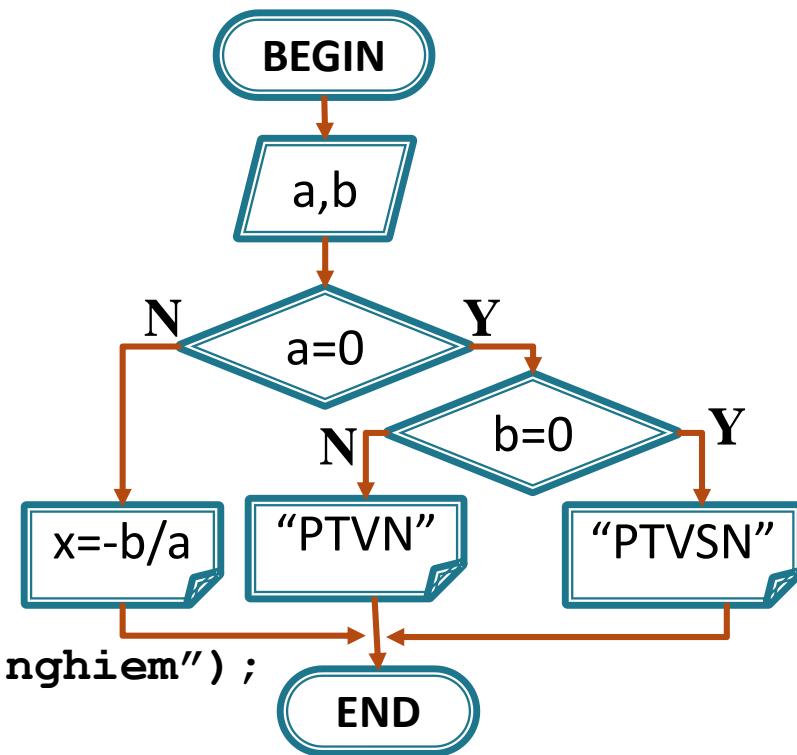
i la so am

3. Cấu trúc rẽ nhánh

3.3. Nhiều lệnh if lồng nhau

Ví dụ: Giải và biện luận phương trình bậc 1: $ax+b=0$

```
void main()
{
    float a, b;
    printf("Nhập vào a: ");
    scanf("%f", &a);
    printf("Nhập vào b: ");
    scanf("%f", &b);
    if (a == 0)
    {
        if (b == 0)
        {
            printf(" Phương trình vô số nghiệm");
        }
        else
        {
            printf(" Phương trình vô nghiệm");
        }
    }
    else
    {
        printf(" Phương trình có nghiệm x=%d", -1.0*b/a);
    }
}
```



3. Cấu trúc rẽ nhánh

3.4. Lưu ý

- Nếu đặt dấu chấm phẩy (;) ở ngay sau biểu thức điều kiện thì khối lệnh đặt trong **if** xem như “**KHÔNG LÀM GÌ**”

Ví dụ:

```
printf("Nhập một số nguyên: ");
scanf("%d", &j);

if(j > 0);
    printf("%d là số dương", j);
```

Kết quả:

Nhập một số nguyên: -6

-6 là số dương

3. Cấu trúc rẽ nhánh

3.5. Biểu thức chọn theo điều kiện

- Cú pháp

(điều kiện) ? BT1 : BT2

- Biểu thức nhận giá trị BT1 nếu điều kiện khác 0 (ĐÚNG), các trường hợp khác nhận giá trị BT2
- Ví dụ

a=5

b=7

if a>b

max=a;

else

max=b;

a=5

b=7

max= a>b? a: b

3. Cấu trúc rẽ nhánh

3.5. Thực hành: đọc chương trình, cho biết kết quả

3.5.1. Bài 1:

```
void main()
{
    int a=9, b=6;
    a++;
    a=a+b--;
    a=a+(--b);
    if (a%2==0)
        printf("a la so chan");
    printf("(a+b)= %d", a+b);
}
```

3. Cấu trúc rẽ nhánh

3.5. Thực hành : đọc chương trình, cho biết kết quả

3.5.2. Bài 2

```
void main()
{
    int a=7, b=8;
    a++;
    if(a<=b)
    {
        a=a+b--;
    }
    else
    {
        --b;
    }
    a--;
    a = (--a)+(--b);
    printf("a=%d, b=%d",a,b);
}
```

<i>a</i>	<i>b</i>	<i>In ra màn hình</i>
7	8	
8	8	
16	7	
15	7	
20	6	
		a=20, b=6

3. Cấu trúc rẽ nhánh

3.6. BÀI TẬP : Viết chương trình

3.6.1. Nhập vào hai số nguyên a, b. In ra màn hình giá trị lớn nhất.

3.6.2. Cho ba số a, b, c nhập vào từ bàn phím. Hãy tìm giá trị lớn nhất của ba số trên và in ra kết quả.

3.6.3. Cho ba số a, b, c nhập vào từ bàn phím. Hãy in ra màn hình theo thứ tự tăng dần các số. Lần lượt thực hiện bài tập trong từng trường hợp:

3.6.3.1. Trường hợp 1: dùng ba biến phụ (max, mid, min)

3.6.3.2. Trường hợp 2: dùng hai biến phụ (max, min)

3.6.4. Viết chương trình nhập vào một số nguyên n gồm ba chữ số. Xuất ra màn hình cho biết chữ số lớn nhất ở vị trí nào?

Ví dụ: $n=291$. Chữ số lớn nhất nằm ở hàng chục (chữ số 9).

3. Cấu trúc rẽ nhánh

3.6. Bài tập viết chương trình

3.6.5. Viết chương trình nhập vào số nguyên n gồm ba chữ số. Xuất ra màn hình theo thứ tự tăng dần của các chữ số. Ví dụ: $n=291$. Xuất ra 129.

3.6.6. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không? In kết quả ra màn hình.

3.6.7. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không? In kết quả ra màn hình.

3.6.8. Viết chương trình thực hiện lần lượt các yêu cầu sau:

- Cho nhập 2 số a và b
- Hỏi người dùng cần thực hiện phép toán nào ($+, -, *, /$)
- Tùy phép tính được chọn, in kết quả tính toán ra màn hình

3. Cấu trúc rẽ nhánh

3.6. Bài tập viết chương trình

3.6.9. Viết chương trình tính tiền cước TAXI. Biết rằng:

- *km đầu tiên là $8000^đ$.*
- *Mỗi km tiếp theo là $7000^đ$.*
- *Nếu lớn hơn 30km thì mỗi km thêm sẽ là $5000^đ$.*

Hãy nhập số km sau đó in ra số tiền phải trả.

3.6.10. Nhập vào 3 số nguyên dương. Kiểm tra xem 3 số đó có lập thành tam giác không? Nếu có hãy cho biết tam giác đó thuộc loại nào? (Cân, vuông, đều, ...).

4. CẤU TRÚC NHIỀU LỰA CHỌN

switch (biểu thức điều kiện)

{

case *const1*:

các câu lệnh ;

break ;

case *const 2*:

các câu lệnh ;

break ;

.....

case *const k*:

<các câu lệnh> ;

break ;

[**default**: các câu lệnh]

}



Giá trị biểu thức điều kiện
(btđk) là số nguyên



Trường hợp giá trị btđk bằng
hằng số **const1**



Trường hợp giá trị btđk bằng
hằng số **const2**



Trường hợp giá trị btđk
không có trong các hằng số
đã liệt kê

4. Câu trúc nhiều lựa chọn

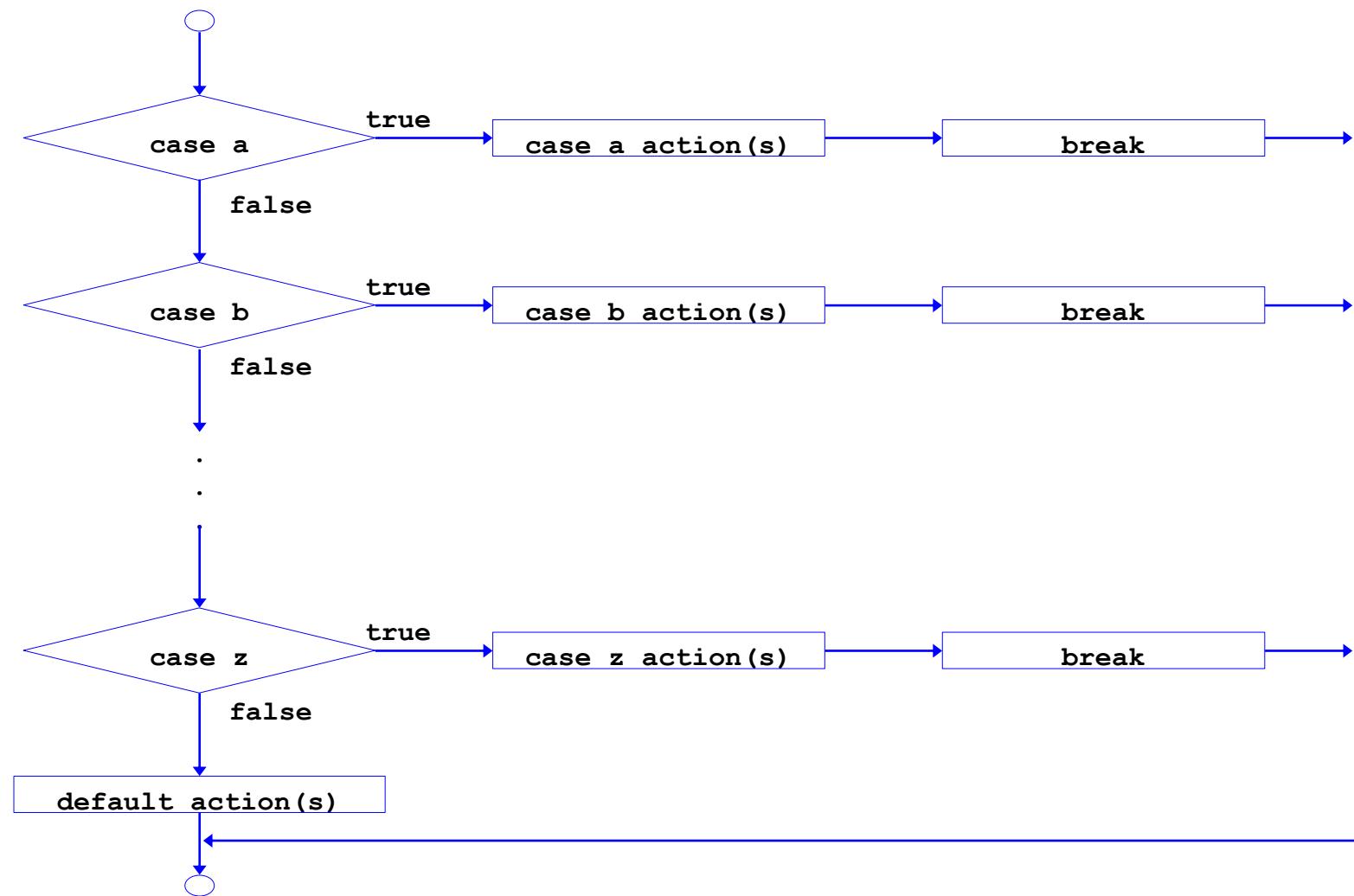
Với:

- **const**, là các **hằng số nguyên hoặc ký tự**.
- Nếu giá trị của biểu thức điều kiện = **const_i**, thì thực hiện câu lệnh sau **case const_i**.
- **default**:
 - Các lệnh đi sau **default** sẽ được thực hiện khi giá trị biểu thức điều kiện không thỏa tất cả các **const_i**.
 - Khi không có sử dụng **default** và không có **const_i** nào được thỏa thì xem như không thực hiện câu lệnh **switch**.
- **break**: Khi chương trình đã thực hiện xong câu lệnh của **case n_i**, nào đó thì nó sẽ thực hiện luôn các lệnh thuộc **case** bên dưới nó mà không xét lại điều kiện (do các **const_i** được xem như các nhãn) → Vì vậy, để chương trình thoát khỏi lệnh **switch** sau khi thực hiện xong một trường hợp, ta dùng lệnh **break**.

```
switch (btđk)
{
    case const1:
        các câu lệnh ;
        break ;
    case const2:
        các câu lệnh ;
        break ;
    .....
    case constk:
        <các câu lệnh> ;
        break ;
    [default: các câu lệnh]
}
```

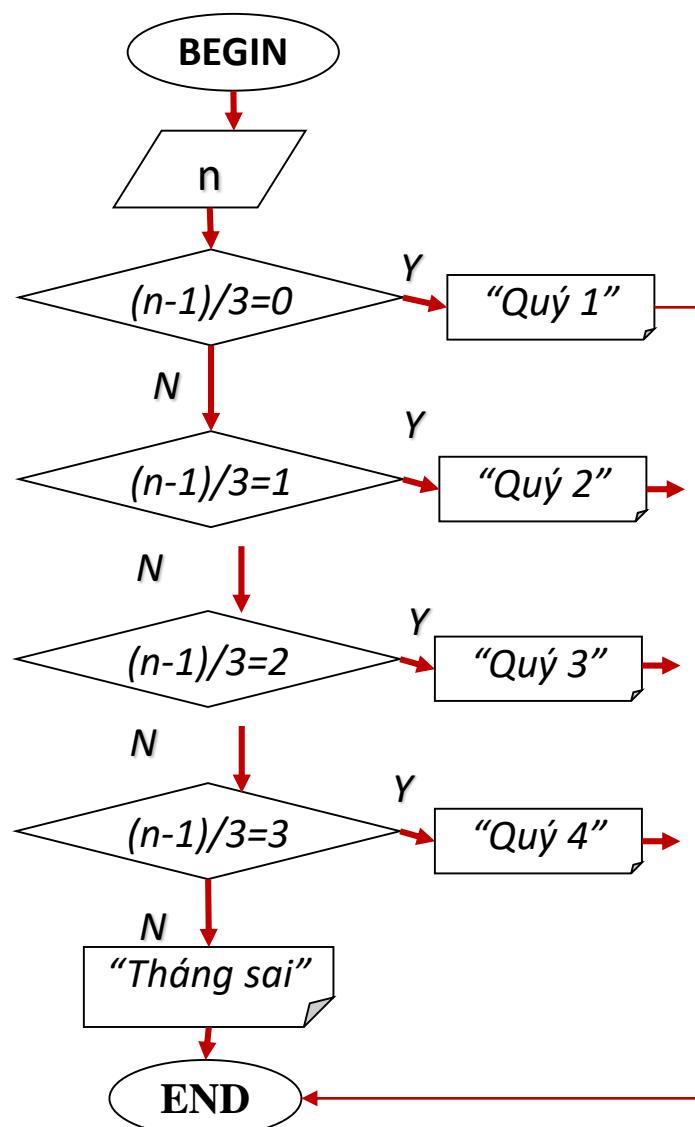
4. Cấu trúc nhiều lựa chọn

Lưu đồ biểu diễn Cấu trúc nhiều lựa chọn – switch



4. Cấu trúc nhiều lựa chọn

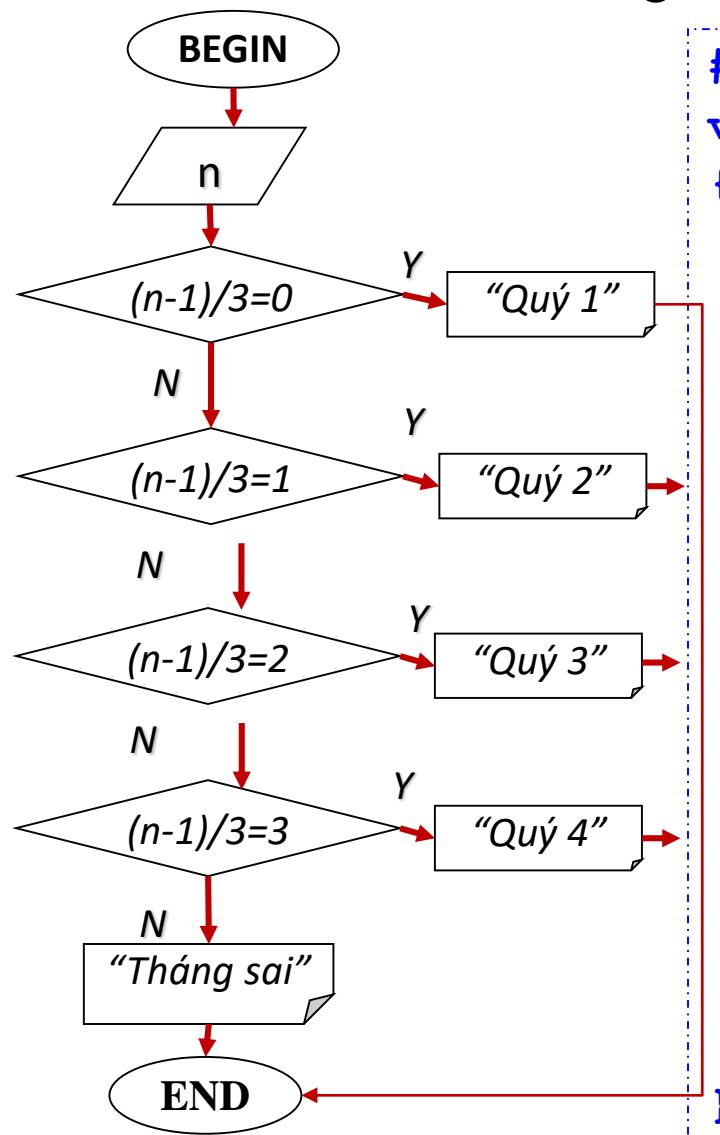
Ví dụ 1: Nhập vào tháng n. Cho biết tháng vừa nhập thuộc quý nào? Vẽ lưu đồ và viết chương trình



```
#include <stdio.h>
void main()
{
    int n;
    printf("Nhập tháng = ");
    scanf("%d", &n);
    x=(n-1)/3;
    if (x == 0)
        printf("Quý 1");
    else
        if (x == 1)
            printf("Quý 2");
        else
            if (x == 2)
                printf("Quý 3");
            else
                if (x == 3)
                    printf("Quý 4");
                else
                    printf("Tháng sai");
}
```

4. Cấu trúc nhiều lựa chọn

Ví dụ 1: Nhập vào tháng n. Cho biết tháng vừa nhập thuộc quý nào? Vẽ lưu đồ và viết chương trình



```
#include <stdio.h>
void main()
{
    int n,x;
    printf("Nhập n = ");
    scanf("%d", &n);
    x=(n-1)/3;
    switch (x)
    {
        case 0: printf("Quý 1");
                  break;
        case 1: printf("Quý 2");
                  break;
        case 2: printf("Quý 3");
                  break;
        case 3: printf("Quý 4");
                  break;
        default: printf("Tháng sai");
    }
}
```

4. Cấu trúc nhiều lựa chọn

Ví dụ 2: Cho nhập 2 số a và b. Nhập phép tính cần thực hiện (+, -, *, /)? Vẽ lưu đồ và viết chương trình

Sử dụng if

```
void main()
{ float a, b, kq;
char c;
printf("Nhập số thu 1:"); scanf("%f", &a);
printf("Nhập số thu 2:"); scanf("%f", &b);
printf("Chọn phép toán (+, -, *, /): ");
c = getchar();
if (!((c=='+') || (c=='-') || (c=='*') || (c=='/')))
{ printf("\nChi nhán các phép toán(+,-,*,/)");
}
else // nhập đúng phép tính
{ if (c == '+')
{ kq = a + b; }
else
{ if (c == '-')
{ kq = a - b; }
else
{ if (c == '*')
{ kq = a * b; }
else
{ if (b!=0)
{ kq = a / b; }
}
}
}
}
if (b == 0)
{ printf("\nMau so phai khac 0");
}
else
{ printf("\nKết quả: %.2f%c%.2f= %.2f", a, c, b, kq); }
}
getch(); }
```

4. Cấu trúc nhiều lựa chọn

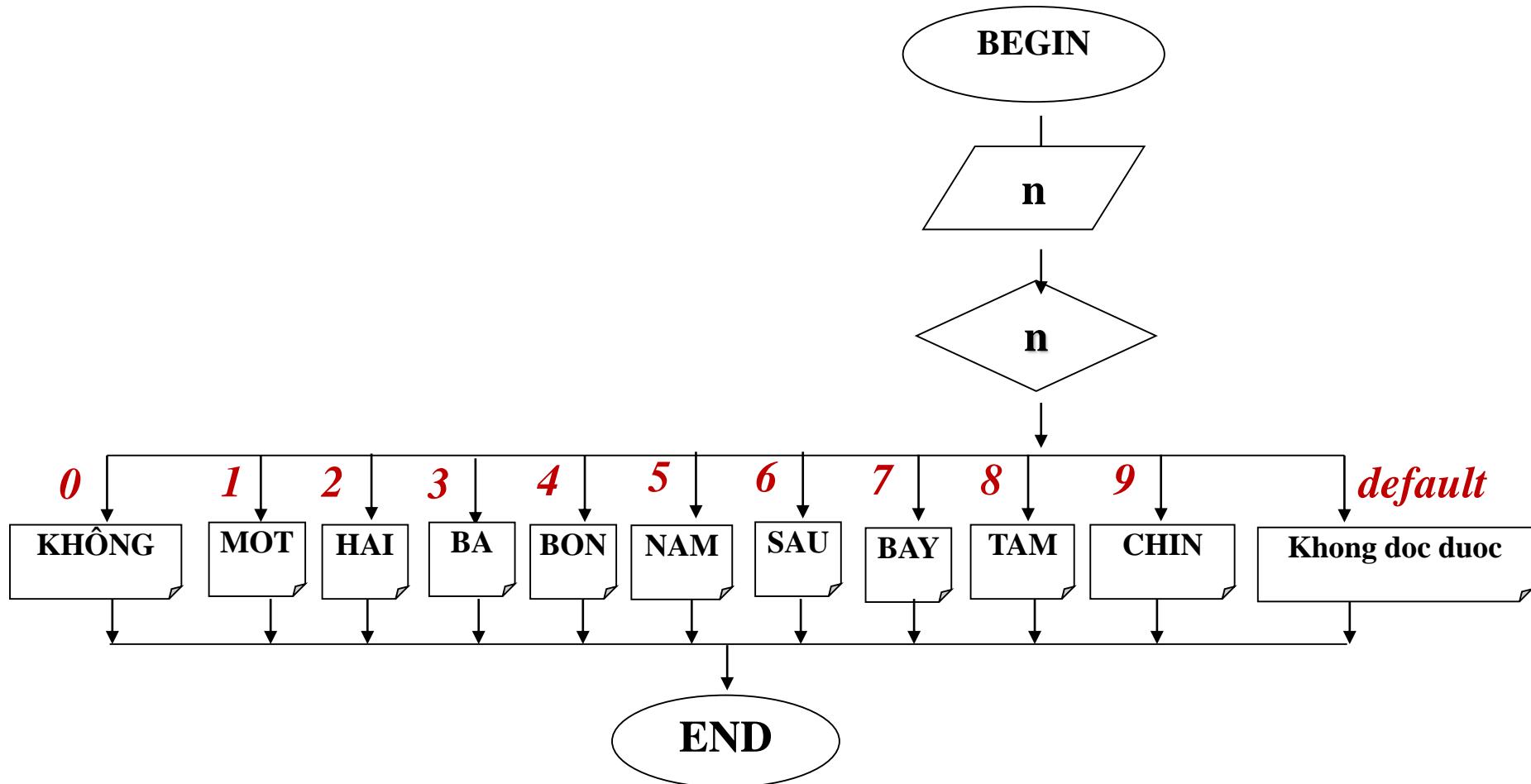
Ví dụ 2: Cho nhập 2 số a và b. Nhập phép tính cần thực hiện (+, -, *, /)? Vẽ lưu đồ và viết chương trình

Sử dụng Switch

```
void main()
{
    float a, b, kq;      char c;
    printf("Nhập số thu 1:"); scanf("%f", &a);
    printf("Nhập số thu 2:"); scanf("%f", &b);
    printf("Chọn phép toán (+, -, *, /): "); c=getchar();
    switch (c)
    {
        case '+':
            printf("\nKết quả: %.2f%c%.2f=% .2f", a, c, b, a+b);
            break;
        case '-':
            printf("\nKết quả: %.2f%c%.2f=% .2f", a, c, b, a-b);
            break;
        case '*':
            printf("\nKết quả: %.2f%c%.2f=% .2f", a, c, b, a*b);
            break;
        case '/':
            if (b == 0)
            {
                printf("\nMau số phải khác 0");
            }
            else
            {
                printf("Kết quả: %.2f%c%.2f=% .2f", a, c, b, a/b);
            }
            break;
        default:
            printf("\nChi nhán trong các phép toán +, -, *, /");
    }
}
```

4. Cấu trúc nhiều lựa chọn

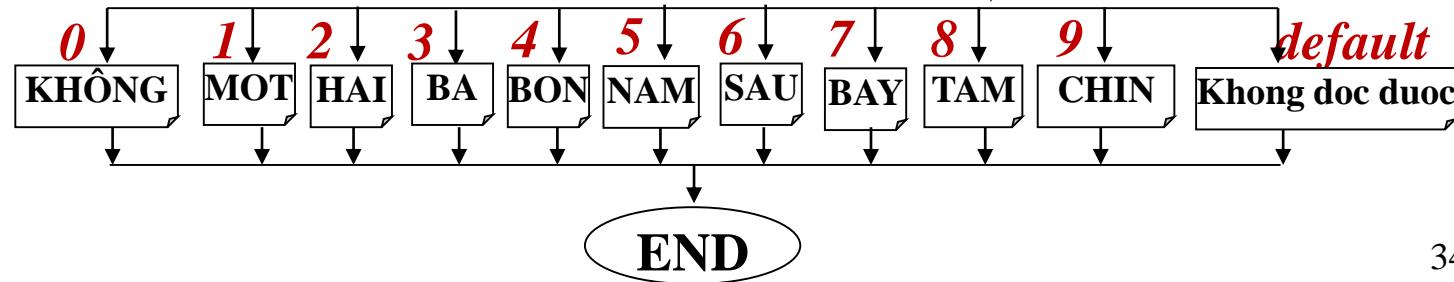
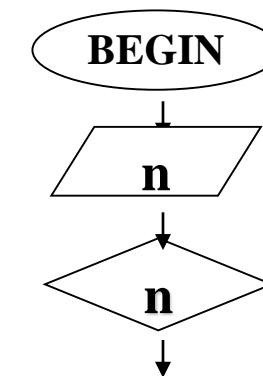
Ví dụ 3: Nhập vào số nguyên n có giá trị từ 0 đến 9. In cách đọc của số đó ra màn hình.



4. Câu trúc nhiều lựa chọn

Ví dụ 3: Nhập số nguyên n (giá trị từ 0 đến 9). In cách đọc của số đó.

```
void main()
{
    int n;
    printf("Nhập vào n (0<=n<=9) : ";
    scanf("%d", &n);
    switch (n)
    {
        case 0: printf("So KHONG"); break;
        case 1: printf("So MOT"); break;
        case 2: printf("So HAI"); break;
        case 3: printf("So BA"); break;
        case 4: printf("So BON"); break;
        case 5: printf("So NAM"); break;
        case 6: printf("So SAU"); break;
        case 7: printf("So BAY"); break;
        case 8: printf("So TAM"); break;
        case 9: printf("So CHIN"); break;
        default: printf("So khong hop le");
    }
}
```



4. Cấu trúc nhiều lựa chọn

Thực hành

Vẽ lưu đồ và viết chương trình cho nhập vào 3 giá trị của ngày, tháng, năm. Kiểm tra xem ngày tháng năm có hợp lệ không? In kết quả kiểm tra ra màn hình.

NỘI DUNG CHƯƠNG

1. Câu lệnh (*statement*) và khối lệnh (*block*)

2. Cấu trúc tuần tự (*Sequence structure*)

Lệnh 1

Lệnh 2

Lệnh 3

...

2. Cấu trúc rẽ nhánh (*Selection structure with one or two branches*)

- if
- if ... else
- if ... else lồng nhau nhiều cấp

3. Cấu trúc lựa chọn (*Selection structure with many branches*) switch

4. Cấu trúc lặp (*Loop structure*)

- for
- while
- do ... while

4. CẤU TRÚC LẶP

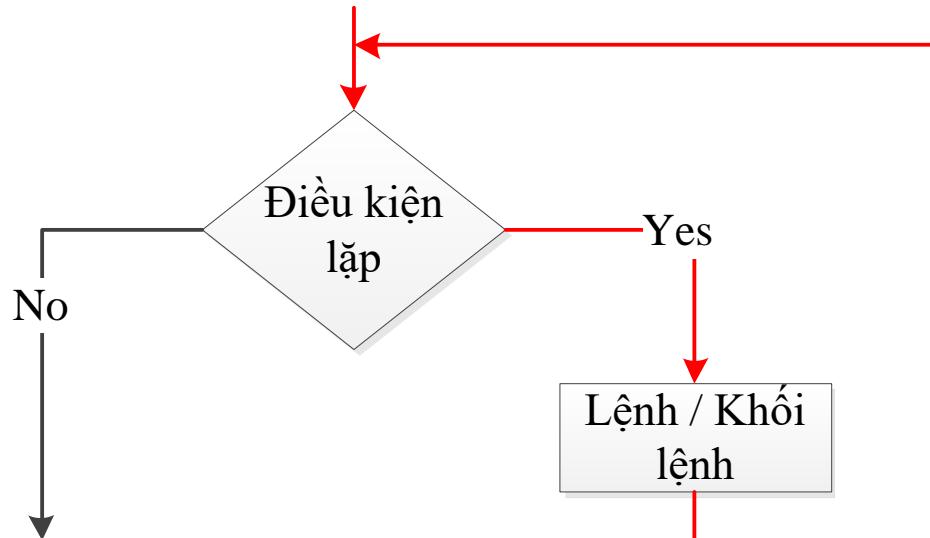
4.1. Đặt vấn đề

- Viết chương trình xuất các số từ 1 đến 1000
 - => Sử dụng 1000 câu lệnh *printf*
- Viết chương trình xuất họ và tên của **n** sinh viên, với n là số lượng do người dùng nhập.
 - => Làm sao xác định giá trị của **n** để có số lượng câu lệnh *printf* cho đúng

Giải pháp:

- Sử dụng “cấu trúc lặp lại” một hành động trong khi còn thỏa một điều kiện nào đó.

4. CẤU TRÚC LẶP



*Kiểm tra điều kiện TRƯỚC
khi lặp*

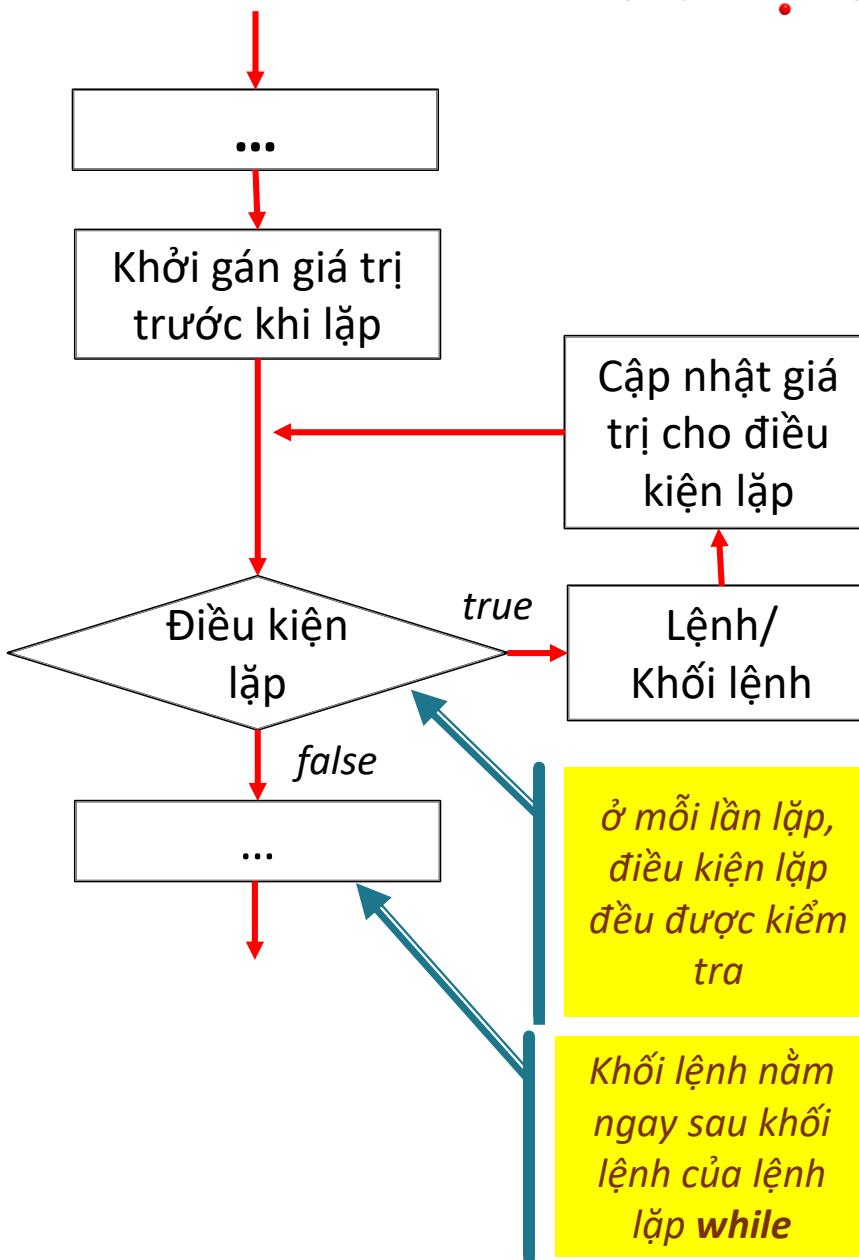
- while
- for



*Kiểm tra điều kiện SAU
khi lặp*

- do . . . while

4.2. LỆNH LẶP while



< Khởi gán >

while (<điều kiện lặp>)

{

- *lệnh/ khói lệnh;*
- *cập nhật giá trị trước khi xét lại điều kiện lặp*

}

Lưu ý:

while thực hiện lệnh/khối lệnh ít nhất 0 lần

4. Cấu trúc lặp

4.2. Lệnh lặp while

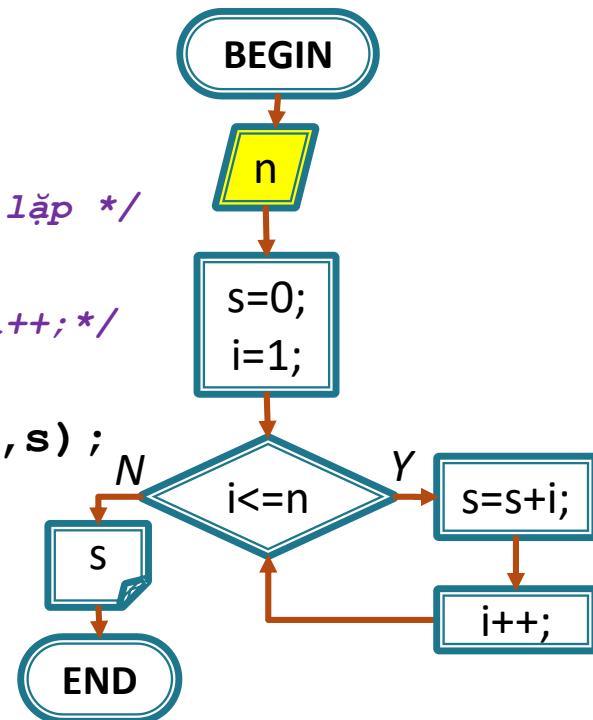
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi gán giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng các số từ 1 đến %d là %d", n, s);
}
```

n=3	s	i



4. Cấu trúc lặp

4.2. Lệnh lặp while

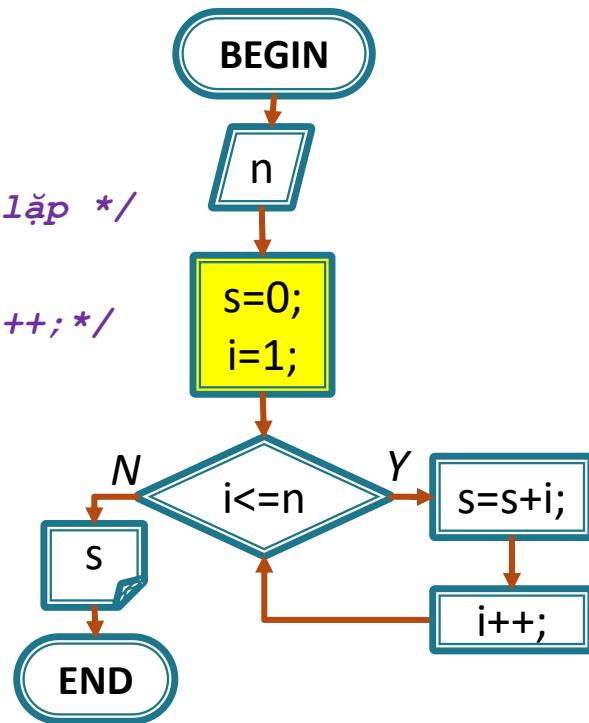
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    /*Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	



4. Cấu trúc lặp

4.2. Lệnh lặp while

Ví dụ 1: Cho nhập số nguyên dương n.

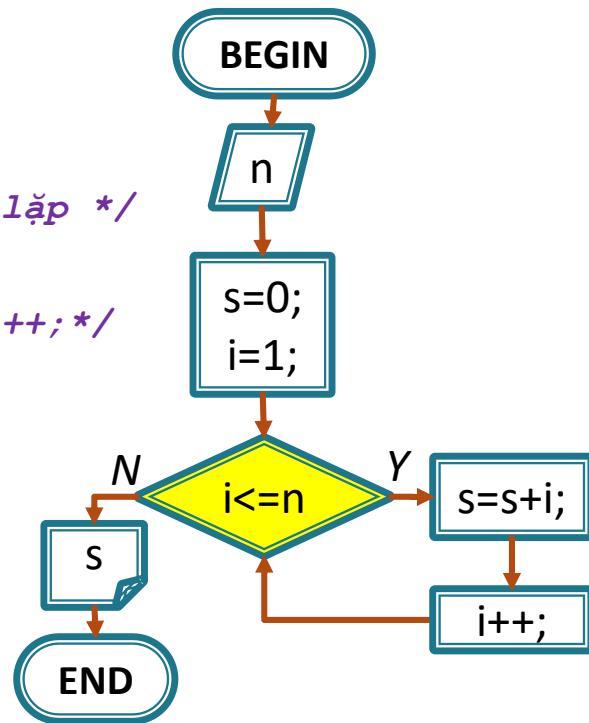
Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;

    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	



4. Cấu trúc lặp

4.2. Lệnh lặp while

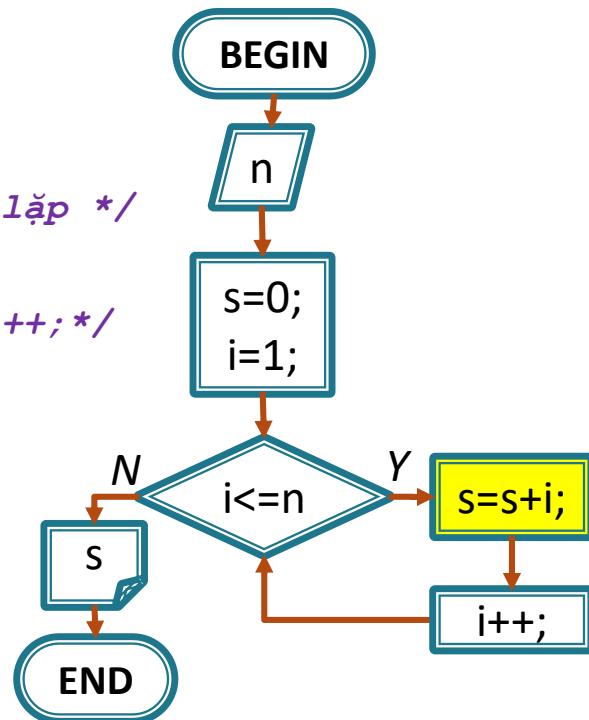
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1



4. Cấu trúc lặp

4.2. Lệnh lặp while

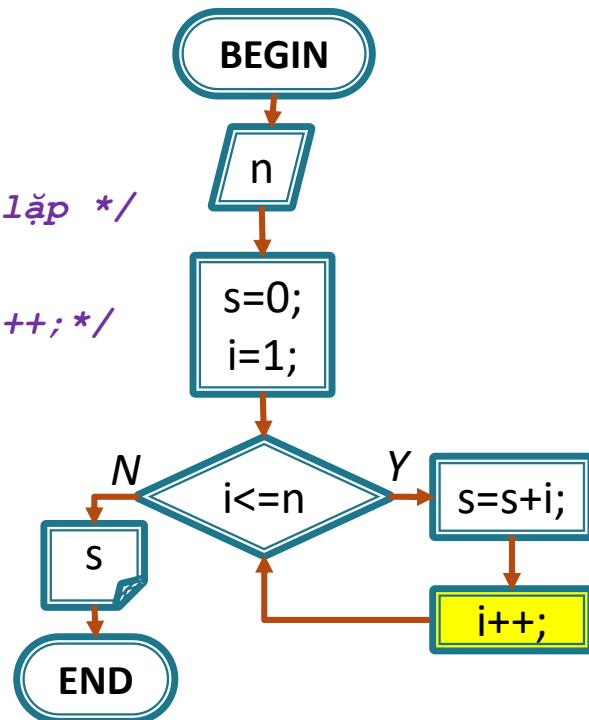
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2



4. Cấu trúc lặp

4.2. Lệnh lặp while

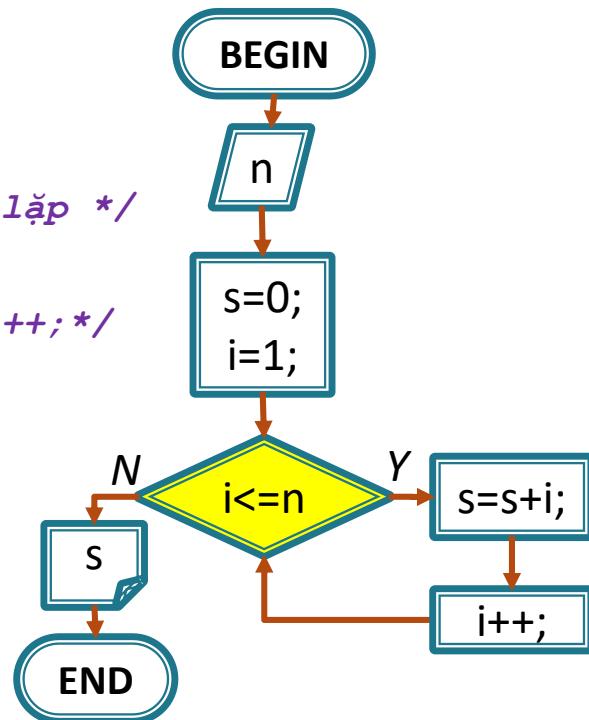
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    /*Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2



4. Cấu trúc lặp

4.2. Lệnh lặp while

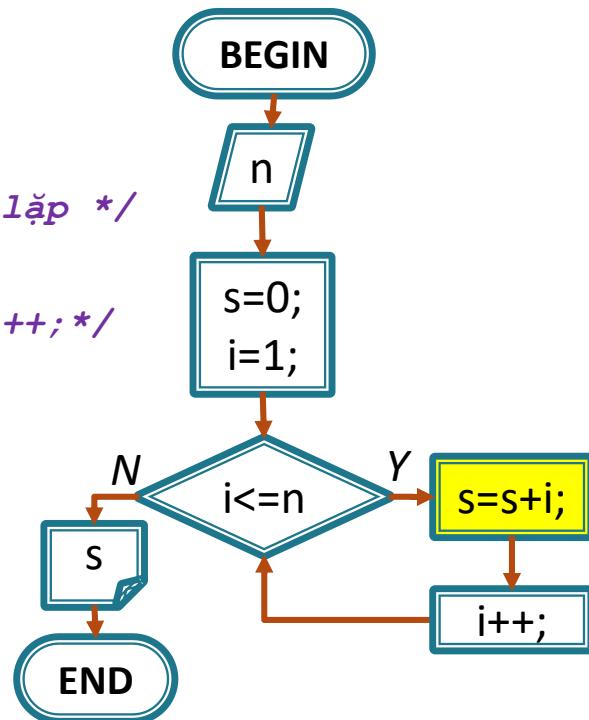
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2



4. Cấu trúc lặp

4.2. Lệnh lặp while

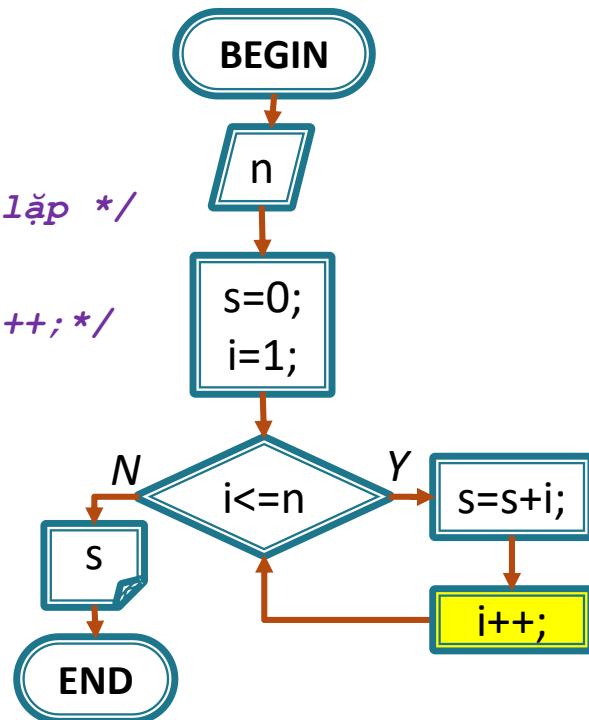
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    /*Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3



4. Cấu trúc lặp

4.2. Lệnh lặp while

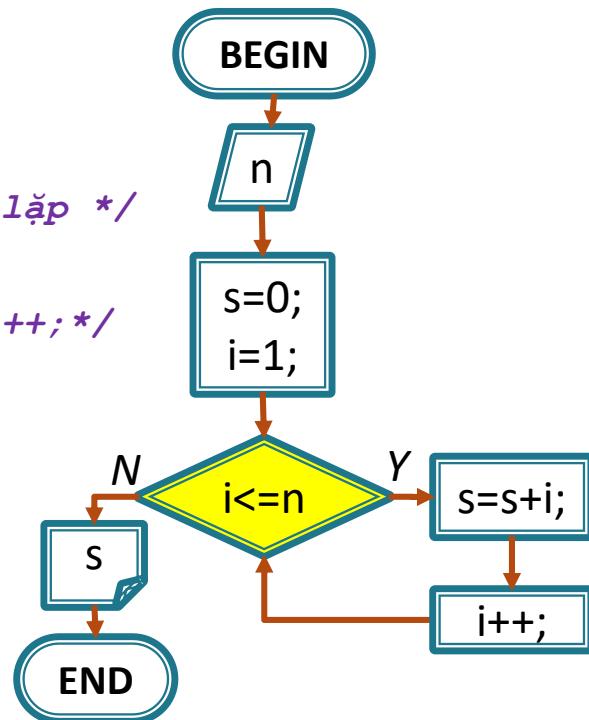
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    /*Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3



4. Cấu trúc lặp

4.2. Lệnh lặp while

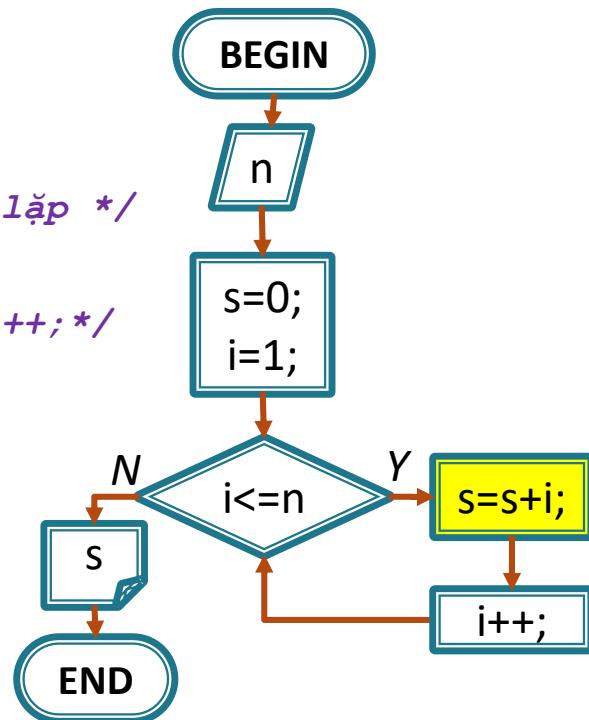
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    /*Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	3



4. Cấu trúc lặp

4.2. Lệnh lặp while

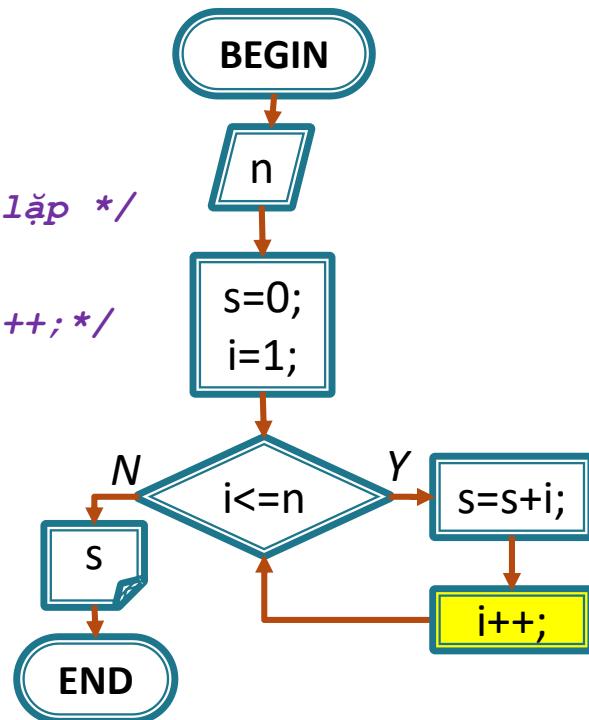
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	4



4. Cấu trúc lặp

4.2. Lệnh lặp while

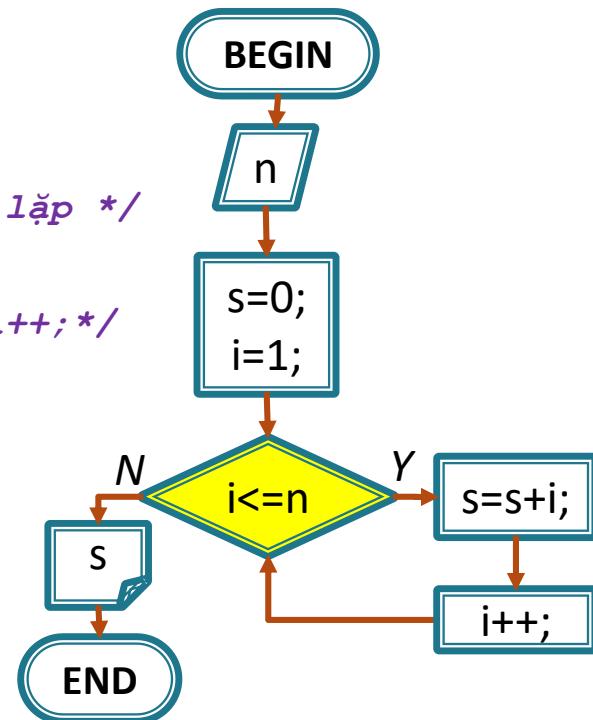
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    /*Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	4



4. Cấu trúc lặp

4.2. Lệnh lặp while

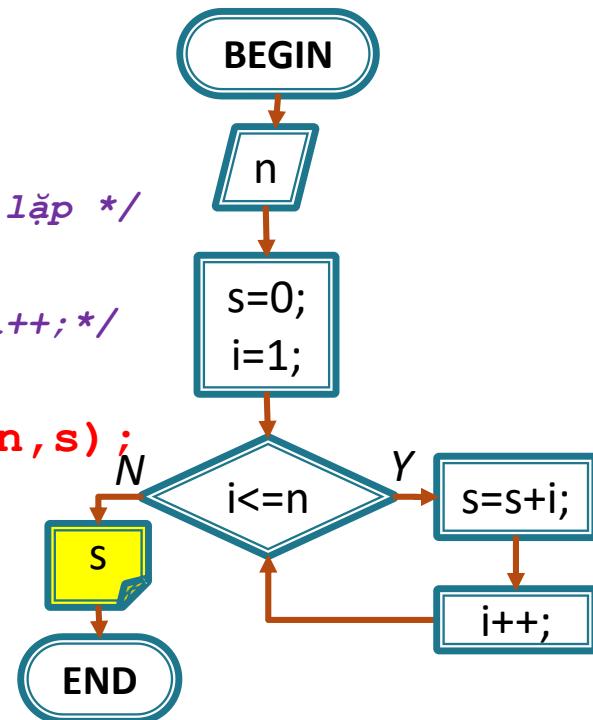
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    while(i<=n)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
        i++;
        /* có thể thay 2 lệnh trên bằng lệnh: s = s + i++;*/
    }
    printf("Tổng các số từ 1 đến %d là %d", n, s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	4



4. Cấu trúc lặp

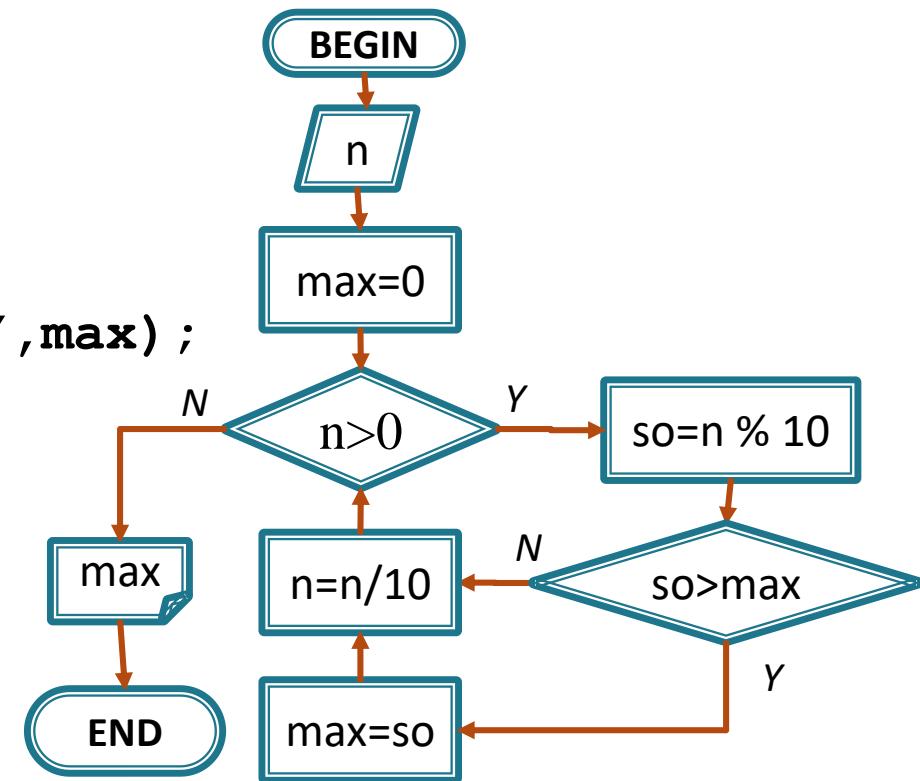
4.2. Lệnh lặp while

Ví dụ 2: Cho nhập số nguyên dương n. In ra số lớn nhất có trong n.

VD $n=5716 \Rightarrow$ in ra **So lon nhat la 7**

```
void main()
```

```
{    int n, so, max=0;  
    printf("Nhập số nguyên dương");  
    scanf("%d", &n);  
    while(n>0)  
    {        so=n%10;  
        if (so>max)  
            max=so;  
        n=n/10  
    }  
    printf("So lon nhat la %d",max);  
}
```



4. Cấu trúc lặp

4.2. Lệnh lặp while

Lưu ý dấu chấm phẩy ;

- Dấu ‘;’ đặt ngay sau biểu thức điều kiện đồng nghĩa với khối lệnh trong while là rỗng. Khi đó khối lệnh đã có (trong while) trở thành đồng cấp và không còn phụ thuộc lệnh while

```
int j = 5;  
  
while(j > 0);  
    printf("j = %i\n", j--);
```

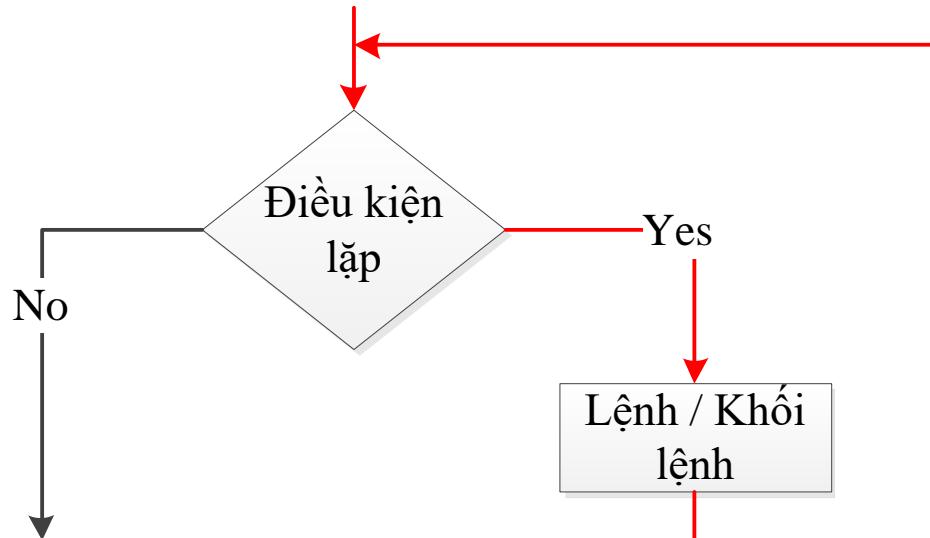
Chương trình bị lặp
không thoát được

- Đôi khi người lập trình cố ý sử dụng **lệnh rỗng**

```
int c, j;  
  
while(scanf("%i", &j) != 1)  
    while((c = getch()) != 27);
```

Đặt dấu ‘;’ để kết thúc
lệnh while. Tức là lệnh
trong while rỗng (không
có các lệnh con)

4. CẤU TRÚC LẶP



*Kiểm tra điều kiện TRƯỚC
khi lặp*

- while
- for



*Kiểm tra điều kiện SAU
khi lặp*

- do . . . while

4. Cấu trúc lặp

4.3. LỆNH LẶP for

- Cách hoạt động của **while** và **for** giống nhau
- Lệnh lặp **for** dồn hết các thành phần của vòng lặp vào trong một câu lệnh.

```
for (<khởi gán>; <điều kiện lặp>; <cập nhật>)
{
    <khối lệnh>;
}
```

- Giải thích
 - **Khởi gán**: Dùng để khởi gán giá trị ban đầu cho vòng lặp
 - **Điều kiện lặp**: Dùng để kiểm tra điều kiện trước khi thực hiện vòng lặp
 - **Cập nhật**: Dùng để cập nhật vòng lặp (tăng hoặc giảm chỉ số lặp)
- Bất kỳ biểu thức nào trong 3 biểu thức nói trên đều có thể vắng nhưng phải giữ dấu chấm phẩy (**;**)



4. Cấu trúc lắp

4.3. Lệnh lặp for

Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
```

```
{ int n, s, i;
```

```
printf("Nhập n: ");
```

```
scanf ("%d", &n) ;
```

//Khởi gán giá trị trước khi lặp

s=0;

$i=1$ to i

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

{ /*Lệnh cần thực hiện yêu cầu của bài toán*/

s=s+i;

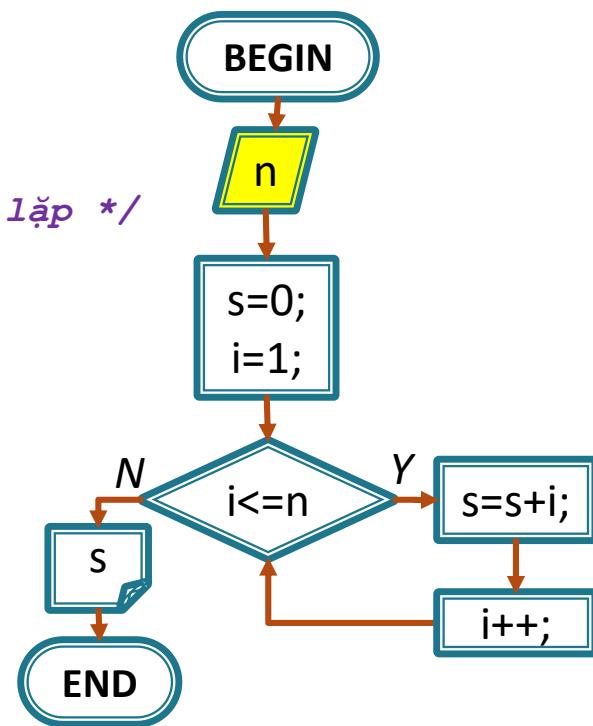
*/*cập nhật giá trị trước khi xét lại điều kiện lặp */*

}

```
printf("Tong=%d", s);
```

}

$n=3$	s	i



4. Cấu trúc lặp

4.3. Lệnh lặp for

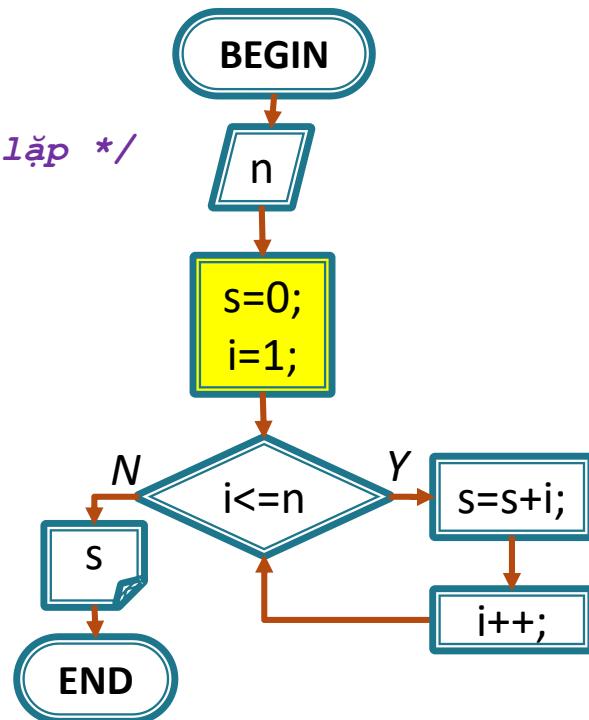
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0		



4. Cấu trúc lặp

4.3. Lệnh lặp for

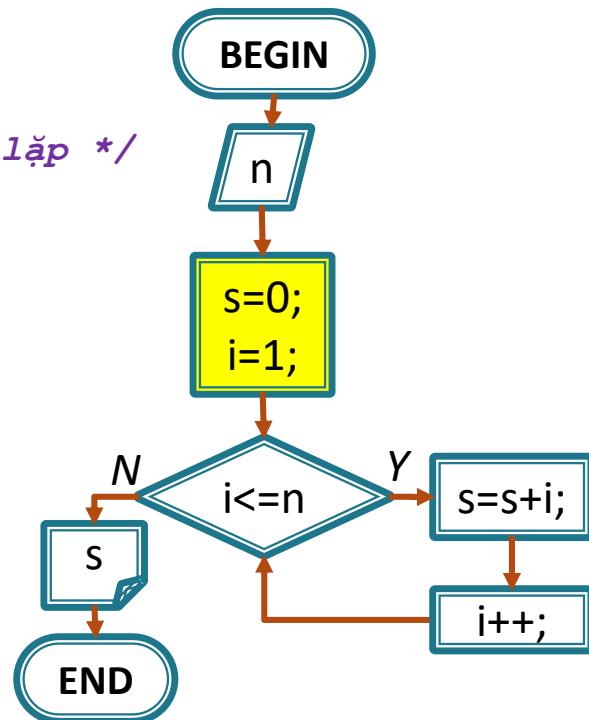
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	



4. Cấu trúc lặp

4.3. Lệnh lặp for

Ví dụ 1: Cho nhập số nguyên dương n.

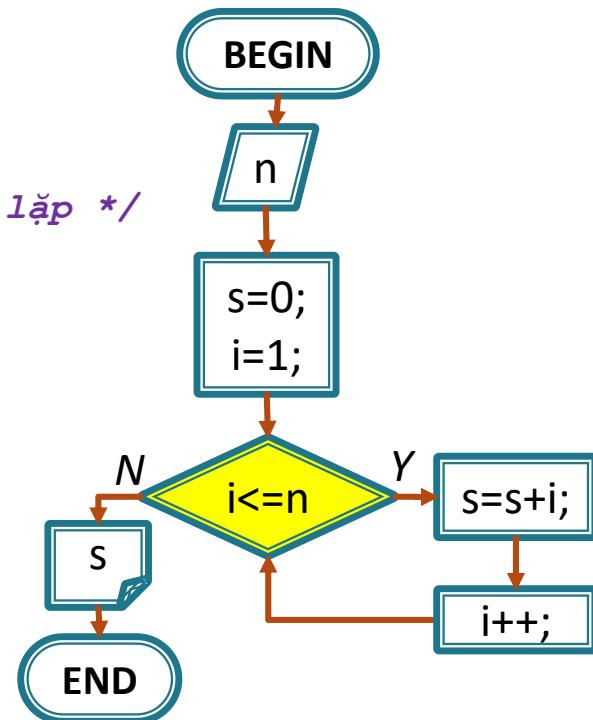
Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;

    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	



4. Cấu trúc lặp

4.3. Lệnh lặp for

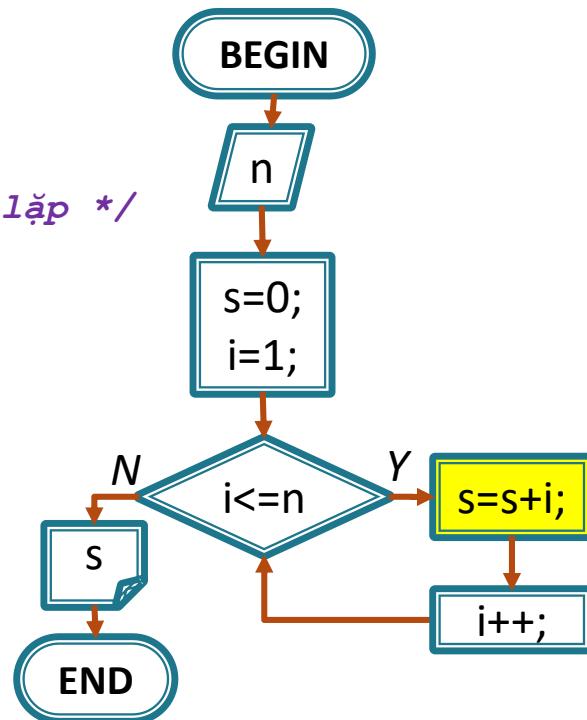
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	
1	1	



4. Cấu trúc lặp

4.3. Lệnh lặp for

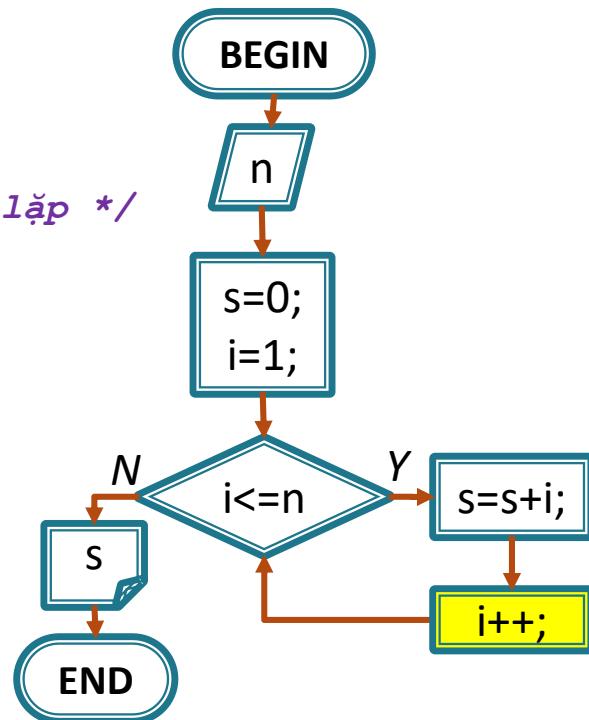
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2



4. Cấu trúc lặp

4.3. Lệnh lặp for

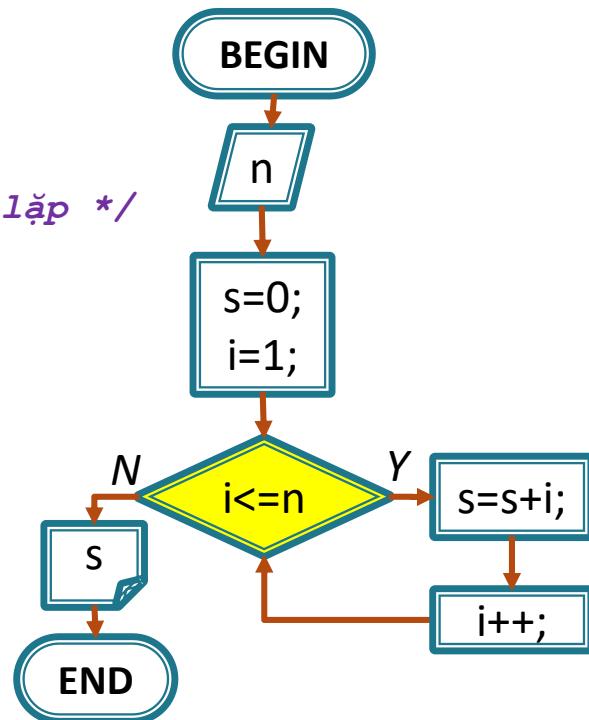
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	
1	1	
1	2	



4. Cấu trúc lặp

4.3. Lệnh lặp for

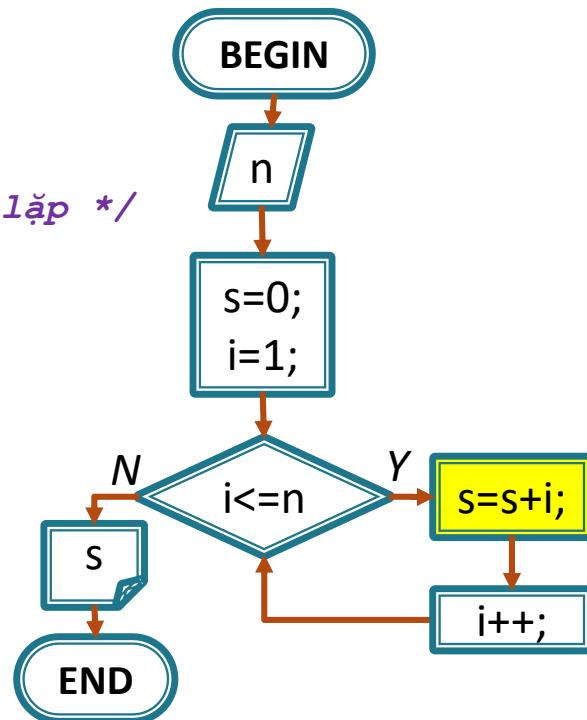
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	
1	1	
1	2	
3	2	



4. Cấu trúc lặp

4.3. Lệnh lặp for

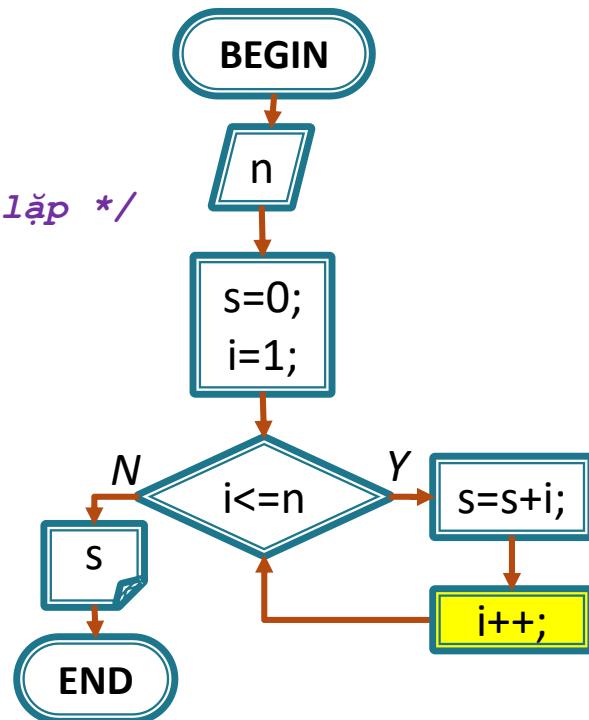
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3



4. Cấu trúc lặp

4.3. Lệnh lặp for

Ví dụ 1: Cho nhập số nguyên dương n.

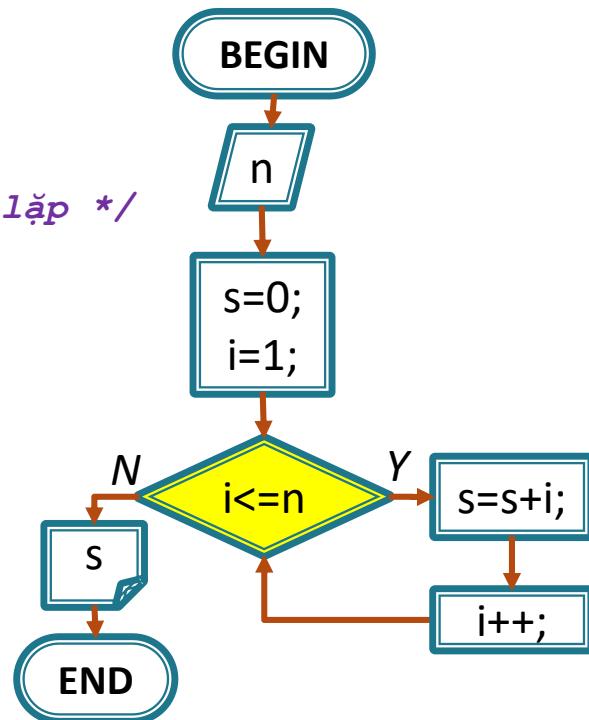
Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;

    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	s	i
0	1	
1	1	
1	2	
3	2	
3	3	



4. Cấu trúc lặp

4.3. Lệnh lặp for

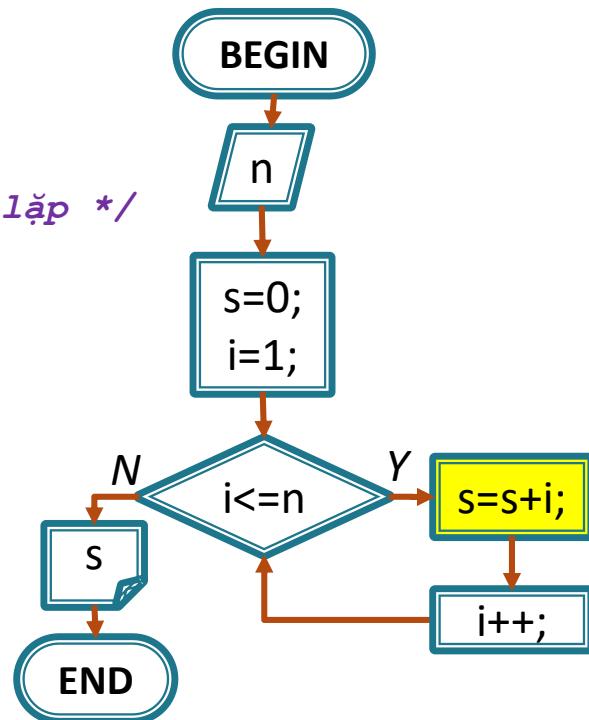
Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);

    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	3



4. Cấu trúc lặp

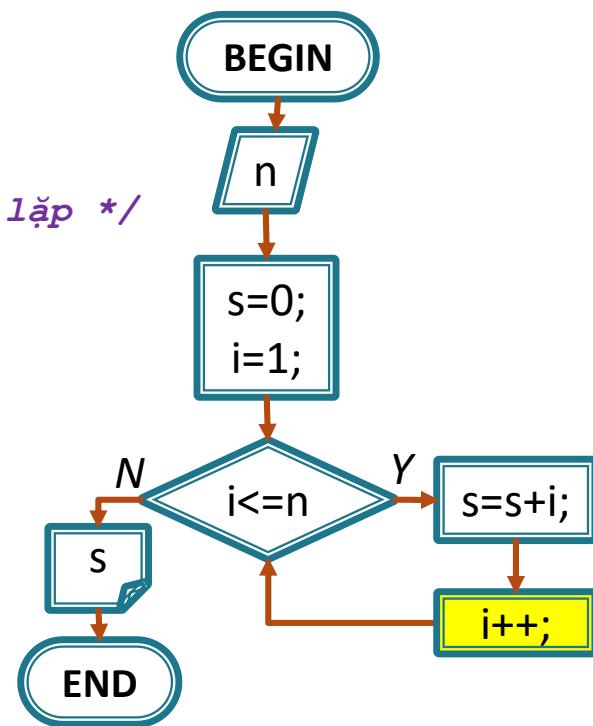
4.3. Lệnh lặp for

Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);
    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	3
6	4



4. Cấu trúc lặp

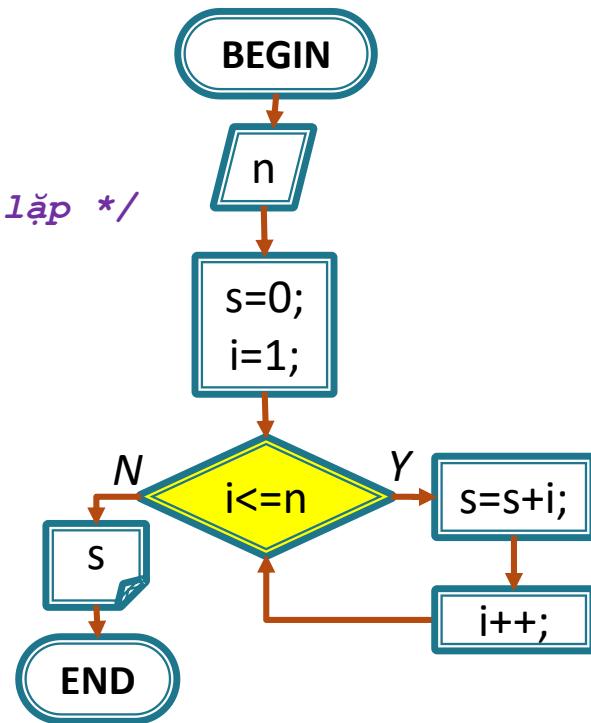
4.3. Lệnh lặp for

Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);
    //Khởi tạo giá trị trước khi lặp
    s=0;
    i=1;
    for(i=1; i<=n; i++)
    {
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
    }
    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	3
6	4



4. Cấu trúc lặp

4.3. Lệnh lặp for

Ví dụ 1: Cho nhập số nguyên dương n.

Tính tổng các số từ 1-n. Ví dụ: n=3

```
void main()
{
    int n, s, i;
    printf("Nhập n: ");
    scanf("%d", &n);
    //Khởi tạo giá trị trước khi lặp
    s=0;

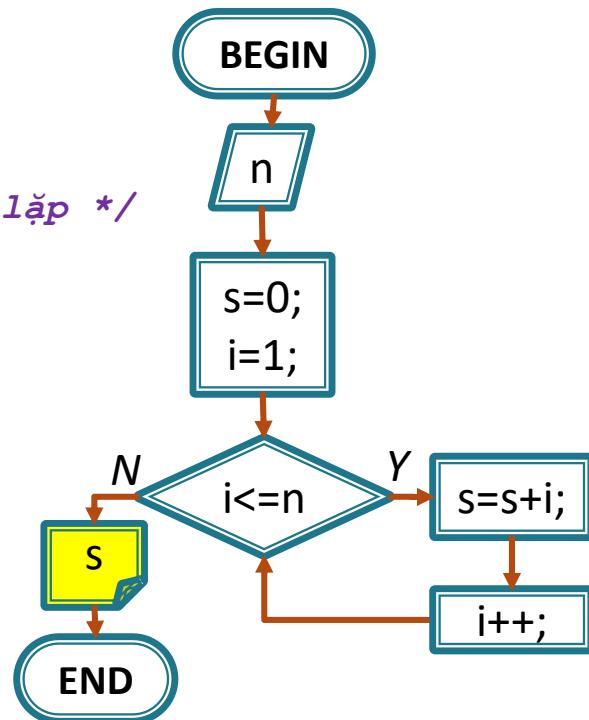
    for(i=1; i<=10; i++)
        /*Lệnh cần thực hiện yêu cầu của bài toán*/
        s=s+i;
        /*cập nhật giá trị trước khi xét lại điều kiện lặp */
}
```

```

    printf("Tổng=%d", s);
}
```

n=3	
s	i
0	1
1	1
1	2
3	2
3	3
6	3
6	4

}



4. Cấu trúc lặp

4.3. Lệnh lặp for

Sử dụng lồng ghép nhiều lệnh trong for

- Phần lệnh khởi động và lệnh điều khiển có thể gồm nhiều lệnh đơn giản, các lệnh này cách nhau bởi dấu phẩy (,)

```
int i, j, k;  
for(i = 0, j = 5, k = -1; i < 10; i++, j++, k--)
```

- Các phần: khởi động, điều kiện lặp, điều khiển có thể không có lệnh nào.

```
for(; i < 10; i++, j++, k--)
```

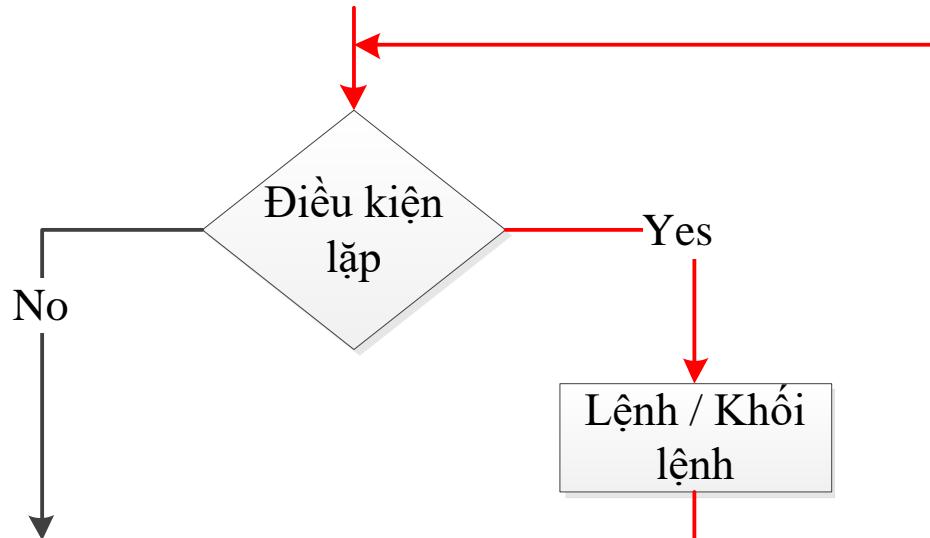
```
for(; i < 10; )
```

Dùng while sẽ hợp lý hơn

```
for(;;)
```

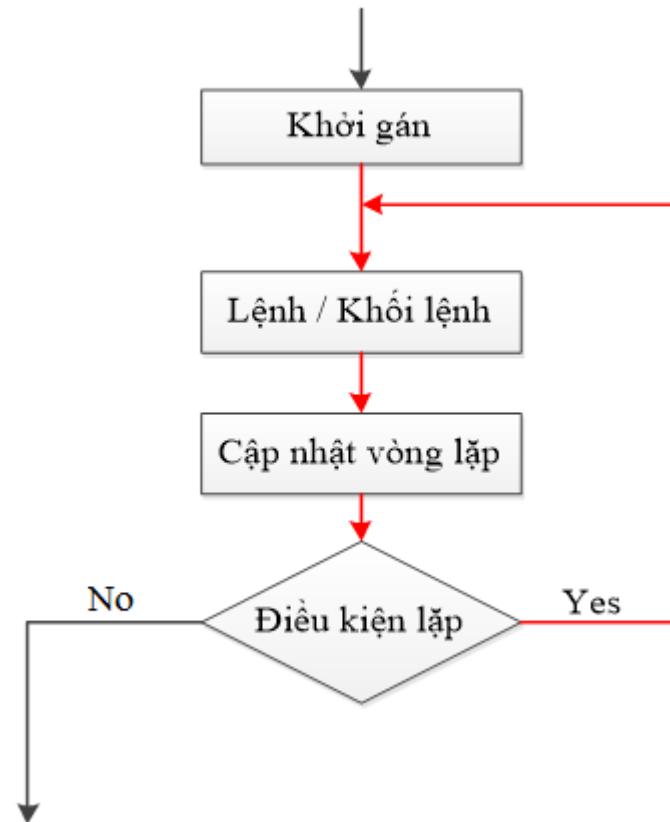
Vòng lặp không kết thúc. Tương đương với việc dùng lệnh **while (1)**

4. CẤU TRÚC LẶP



*Kiểm tra điều kiện TRƯỚC
khi lặp*

- while
- for



*Kiểm tra điều kiện SAU
khi lặp*

- do . . . while

4. Cấu trúc lặp

4.4. LỆNH do ... while

< Khởi gán>

do
{

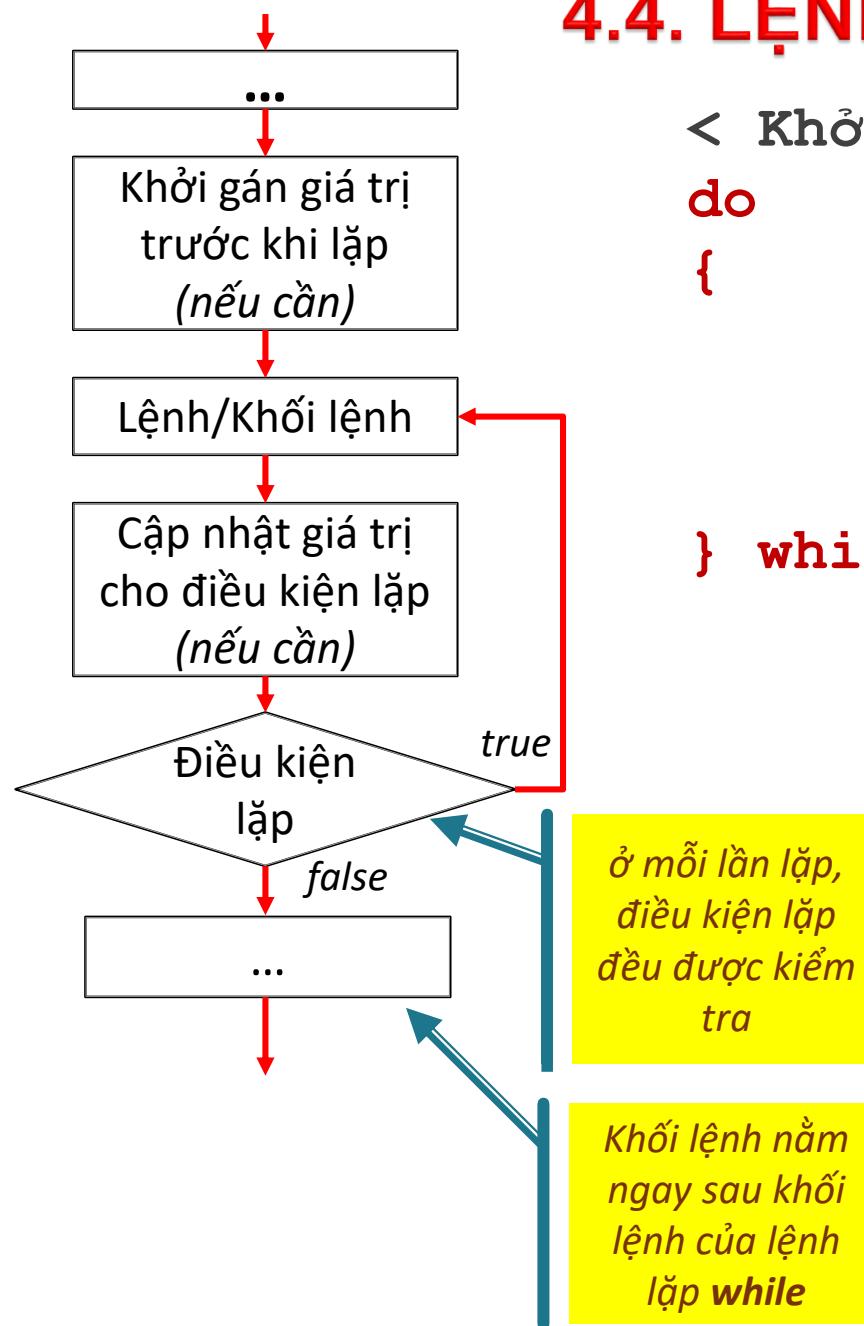
<khối lệnh>;

<cập nhật>;

} while (điều kiện);

☞ Thực hiện khối lệnh cho đến khi biểu thức điều kiện có giá trị bằng false.

☞ do while thực hiện lệnh S ít nhất 1 lần. Do đó, cấu trúc lặp do...while thường dùng cho trường hợp cho người dùng nhập dữ liệu trước khi chương trình thực hiện kiểm tra điều kiện



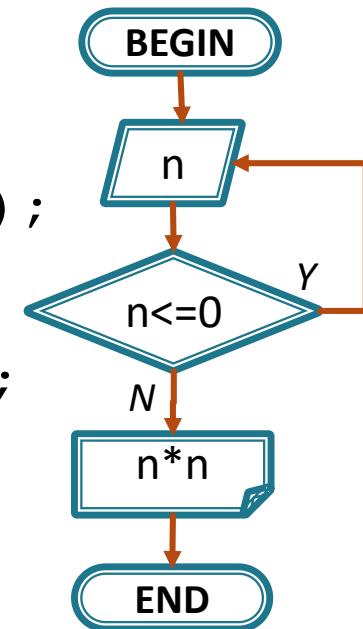
4. Cấu trúc lặp

4.4. Lệnh lặp do ... while

Ví dụ: Nhập vào một số nguyên dương n ($n > 0$), nếu nhập sai thì yêu cầu nhập lại. Nếu nhập đúng in ra giá trị của n^2

// thay vì viết

```
void main()
{ int n;
  do
  { printf("Nhập vào một số nguyên dương: ");
    scanf("%d", &n);
  } while (n<=0);
  printf("Bình Phuong cua %d la %d", n, n*n);
}
// nên viết lại là
void main()
{ int n;
  do
  { printf("Nhập vào một số nguyên dương: ");
    scanf("%d", &n);
  } while (!(n>0));
  printf("Bình Phuong cua %d la %d", n, n*n);
}
```



4. Cấu trúc lặp

4.4. Lệnh lặp do ... while

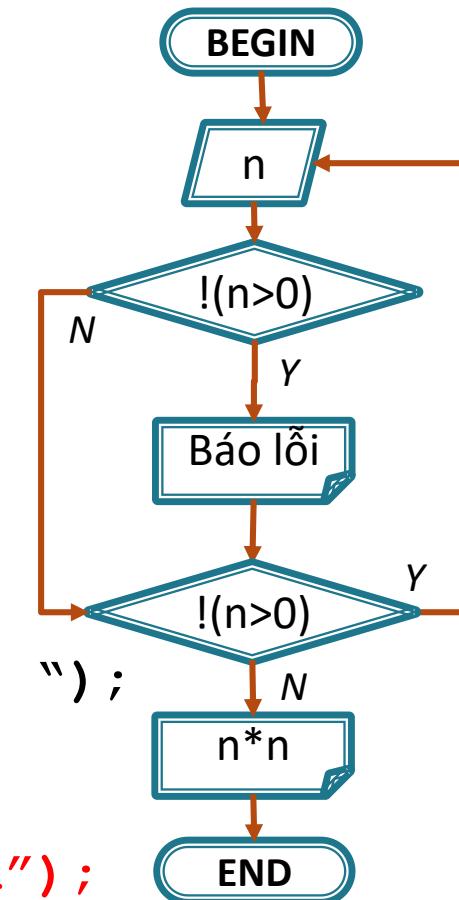
Ví dụ: Nhập vào một số nguyên dương n ($n > 0$), nếu nhập sai thì yêu cầu nhập lại. Nếu nhập đúng in ra giá trị của n^2

// thay vì viết

```
void main()
{ int n;
do
{ printf("Nhập vào một số nguyên dương: ");
  scanf("%d", &n);
} while (! (n>0));
printf("Bình Phuông của %d là %d", n, n*n);
}
```

// Mở rộng

```
void main()
{ int n;
do
{ printf("Nhập vào một số nguyên dương: ");
  scanf("%d", &n);
  if (! (n>0))
    printf("Nhập sai, hãy nhập lại!\n");
} while (! (n>0));
printf("Bình Phuông của %d là %d", n, n*n);
}
```



4.5. LỆNH **break** và **return**

- Lệnh **break**: thoát khỏi các cấu trúc *switch*, *while*, *for*, *do...while* trong cùng chung chứa lệnh **break**. Tại thời điểm **break** được gọi thi hành, tất cả các lệnh đi ngay sau **break** trong cùng vòng lặp sẽ không thực hiện (kể cả việc kiểm tra kết quả của biểu thức điều kiện – nếu có).
- Lệnh **return**: là lệnh trả về từ hàm, nghĩa là thoát khỏi hàm đang thi hành, nên cũng giúp thoát luôn khỏi tất cả các vòng lặp.

4.6. LỆNH continue

- Được sử dụng trong các vòng lặp như **while**, **for**, **do...while**.
- Khi lệnh **continue** được gọi, chương trình sẽ quay trở về đầu vòng lặp để bắt đầu lần lặp mới (có kiểm tra điều kiện lặp để xác định có lặp tiếp hay không). Khi đó, nếu có các lệnh còn lại (cùng trong vòng lặp) đặt sau **continue** sẽ không được thực hiện.

MỘT SỐ VÍ DỤ VỀ LỆNH break, return và continue

Ví dụ: Cho phép người dùng nhập số nguyên dương, nếu nhập đúng thì dừng, nếu sai cho nhập lại

```
void main()
{
    int n;
    while (1)
    {
        printf("Nhập giá trị nguyên dương:");
        scanf("%d", &n);
        if (n >= 0)
        {
            printf("Số vừa nhập hợp lệ");
            break;
        }
        else
            printf("Phải nhập số dương");
    }
}
```

Một số ví dụ về lệnh break, return và continue

Ví dụ: Cho phép người dùng nhập số nguyên dương, nếu nhập đúng thì dừng, nếu sai cho nhập lại

```
void main()
{
    ...
    int code;
    while(1)-----.
    {
        printf("Hay cho biet mat ma: ");
        scanf("%d", &code);
        if (code == 0)      return; //kết thúc hàm main -----
        if (code != 999)   continue; //quay lại lệnh while(1) -----.

        break; // nhập đúng, kết thúc lệnh while(1) -----
    } //end while
    printf("Code=%d", code); -----
} //end main
-----.
```

4. Cấu trúc lắp

MỘT SỐ VÍ DỤ VỀ LỆNH break và continue (tt)

Ví dụ: In ra màn hình giá trị từ 10 đến 20 trừ đi số 13 và số 17.

```
void main()
{
    for (int k = 10; k <= 20; k++)
    {
        if ( (k == 13) || (k == 17) )
            continue;
        printf("%d\t", k);
    }
}
```

4.7. CHUYỂN ĐỔI GIỮA CÁC LỆNH LẬP

4.7.1. Chuyển đổi giữa for và while

Ví dụ: In ra màn hình giá trị từ 10 đến 20.

```
void main()
{
    for (int k=10; (k<=20); k++)
    {
        printf("%5d", k);
    }
}
```

```
void main()
{
    while
    {
    }
}
```

4.7. CHUYỂN ĐỔI GIỮA CÁC LỆNH LẶP

4.7.1. Chuyển đổi giữa for và while

Khi nào nên dùng while? Khi nào nên dùng for?

Ví dụ: Cho người dùng nhập nhiều số. Tính tổng các số đã nhập. Việc nhập kết thúc khi người dùng nhập số 0

```
void main()
{
    int s=0, so=1;
    for (; so!=0;)
    {
        printf("Nhập số: ");
        scanf("%d", &so);
        s=s+so;
    }
    printf("S=%5d", s);
}
```

```
void main()
{
    int s=0, so=1;
    while (so!=0)
    {
        printf("Nhập số: ");
        scanf("%d", &so);
        s=s+so;
    }
    printf("S=%5d", s);
}
```

4. Cấu trúc lặp

4.7. Chuyển đổi giữa các lệnh lặp

4.7.2. Chuyển đổi giữa for và do ... while

Ví dụ: In ra màn hình giá trị từ 10 đến 20 ngoại trừ số 15 .

```
void main()
{
    for (int k=10; (k<=20); k++)
    {
        if (k == 15)
            continue;
        printf("%d\t");
    }
}
```

```
void main()
{
    do
    {
        while
    } while
};
```

4. Cấu trúc lặp

4.7. Chuyển đổi giữa các lệnh lặp

4.7.3. Chuyển đổi giữa while và do ... while

Ví dụ: In ra màn hình giá trị từ 10 đến 20 ngoại trừ số 15 .

```
void main()
{   int k=10;
    while (k<=20)
    {
        if (k == 15)
            continue;
        printf("%d\t");
        k++;
    }
}
```

```
void main()
{
    do
    {
        printf("%d\t");
        k++;
    } while (k<=20);
}
```

4. Cấu trúc lặp

4.8. Một số cách dùng lệnh FOR

```
for (initializationStatement; testExpression; updateStatement)  
{  
    // statements inside the body of loop  
}
```

Ví dụ: Tính $S = 1 + (1+2) + (1+2+3) + (1+2+3+4) + \dots + (1+2+3+\dots+n)$

4.8.1. Chuyển tham số 1 và 3 của for

```
int Tong(int n)  
{  
    int S = 0, sPre=0;  
    for (int i = 1; i <= n; i++)  
    {  
        sPre += i;  
        S += sPre;  
    }  
    return S;  
}
```



```
int Tong(int n)  
{  
    int S = 0, sPre=0;  
    int i = 1;  
    for (; i <= n;)  
    {  
        sPre += i;  
        S += sPre;  
        i++;  
    }  
    return S;  
}
```

4. Cấu trúc lặp

4.8. Một số cách dùng lệnh FOR

```
for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}
```

Ví dụ: Tính $S = 1 + (1+2) + (1+2+3) + (1+2+3+4) + \dots + (1+2+3+\dots+n)$

4.8.2. Chuyển cả 3 tham số của for

```
int Tong(int n)
{
    int S = 0, sPre=0;
    int i = 1;
    for (; i <= n;)
    {
        sPre += i;
        S += sPre;
        i++;
    }
    return S;
}
```



```
int Tong(int n)
{
    int S = 0, sPre=0;
    int i = 1;
    for (;;)
    {
        if (i <= n)
        {
            sPre += i;
            S += sPre;
        }
        i++;
    }
    return S;
}
```

4. Cấu trúc lặp

4.8. Một số cách dùng lệnh FOR

```
for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}
```

Ví dụ: Tính $S = 1 + (1+2) + (1+2+3) + (1+2+3+4) + \dots + (1+2+3+\dots+n)$

4.8.3. Chuyển tất cả các lệnh trong thân của for

```
int Tong(int n)
{
    int S = 0, sPre=0;
    for (int i=1;i<=n;i++)
    {
        sPre += i;
        S += sPre;
    }
    return S;
}
```



```
int Tong(int n)
{
    int S = 0, sPre=0;
    for (int i=1;i<=n;sPre+=i,S+=sPre,i++);
    return S;
}
```

4. Cấu trúc lặp

4.8. Một số cách dùng lệnh FOR

Ví dụ: Tính $S = 1 + (1+2) + (1+2+3) + \dots + (1+2+3+\dots+n)$

4.8.4. Thực hành

```
int Tong(int n)
{
    int S = 0 ;
    for (int i=1;i<=n; i++)
    {
        int s1=0;
        for (int j=1;j<=i ;j++)
            s1 += j;
        S += s1;
    }
    return S;
}
```

```
int Tong(int n)
{
    int S=0, s1=0, i = 1 ;
    for (; i <= n; S += s1, s1 = 0, i++)
        for (int j = 1; j <= i; s1 += j, j++);
    return S;
}
```



4.8. Một số cách dùng lệnh FOR

4.8.4. Thực hành: viết hàm nhận tham số là số nguyên n. In ra n đảo ngược. Ví dụ: n=12345 \Rightarrow xuất ra màn hình 54321

```
void InNguoc(int n)
{
    int so;
    for (; n > 0; n /= 10)
    {
        so = n % 10, n /= 10,
        printf("%d", so);
    }
}
```

```
void InNguoc(int n)
{
    for (int so; n > 0; so = n % 10, printf("%d", so) , n /= 10);
}
```



4.8. BÀI TẬP

4.8.1. XÁC ĐỊNH KẾT QUẢ

4.8.1.1.

```
int a=18;  
for(int i = 1; i <= a; i++)  
    if(a%i == 0)  
        printf("%d\t",i);
```

4.8.1.2.

```
for(int i = 0; i < 5; i++)  
{  
    for(int j = 0; j <= i; j++)  
        printf("%d\t",j);  
    printf("\n",j);  
}
```

4. Cấu trúc lặp

4.8. Bài tập

4.8.1. Xác định kết quả

4.8.1.3.

```
int i = 10, s = 0;  
while(i > 0)  
{  
    if(i%2 == 0)  
        s+=i;  
    else  
        if(i > 5)  
            s+=2*i;  
    i--;  
}  
printf("s = %d", s);
```

4. Cấu trúc lặp

4.8. Bài tập

4.8.1. Xác định kết quả

4.8.1.4.

```
int a = 18, i = 1;  
do{  
    if(a%i == 0)  
        printf("%d\t",i);  
    i++;  
} while(i <= a);
```

4.8.1.5.

```
int a = 11, b = 16, i = a;  
while( i < b )  
{    if(i%2 == 0)  
    {        printf("%d\t",i);  
        break;  
    }  
    i++;  
}
```

4. Cấu trúc lặp

4.8. Bài tập

4.8.1. Xác định kết quả

4.8.1.6.

```
int a = 10, s = 0, i = 0;  
while( i < a )  
{  
    i++;  
    if (i % 2 == 0)  
        continue;  
    else  
        s=s+i;  
}  
printf("s=%d",s);
```

4. Cấu trúc lặp

4.8. Bài tập

4.8.1. Xác định kết quả

4.8.1.7.

```
int i = 1, s = 0;  
while(1)  
{  
    s = s + i++;  
    if (i % 2)  
        i = i + 2;  
    else  
        i = i + 1;  
    if (i > 20)  
        break;  
}  
printf("s = %d",s);
```

4. Cấu trúc lập

4.8. Bài tập

4.8.2. Viết chương trình

4.8.2.1. Cho nhập số nguyên n. Xét xem n có phải là số nguyên tố hay không?

4.8.2.2. Viết chương trình nhập số nguyên dương n. Liệt kê n số nguyên tố đầu tiên.

4.8.2.3. Viết chương trình nhập vào hai số nguyên dương a và b. Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của a và b.

4.8.2.4. Viết chương trình nhập vào một số nguyên n gồm tối đa 10 chữ số (4 bytes). In ra màn hình giá trị nhị phân của số trên. (Hướng dẫn: chia lấy dư cho 2 và xuất theo thứ tự ngược lại).

4. Cấu trúc lặp

4.8. Bài tập

4.8.2. Viết chương trình

4.8.2.5. Viết chương trình đếm số ước số của số nguyên dương N.

Ví dụ: $N=12$ số ước số của 12 là 6

4.8.2.6. Một số hoàn thiện là một số có tổng các ước số của nó (không kể nó) bằng chính nó. Hãy liệt kê các số hoàn thiện nhỏ hơn 30000 .

Ví dụ: số 6 là số hoàn thiện vì tổng các ước số là $1+2+3 = 6$.

4.8.2.7. Nhập vào ngày, tháng, năm. Cho biết đó là ngày thứ mấy trong năm.

4.8.2.8. In ra dãy số Fibonacci

- $f_1 = f_0 = 1;$
- $f_n = f_{n-1} + f_{n-2}; \quad (n > 1)$