

# Nhập Môn Lập Trình

## Cấu Trúc Vòng Lặp

TS. Lê Nguyên Khôi

Trường Đại học Công nghệ, ĐHQGHN

# Nội Dung

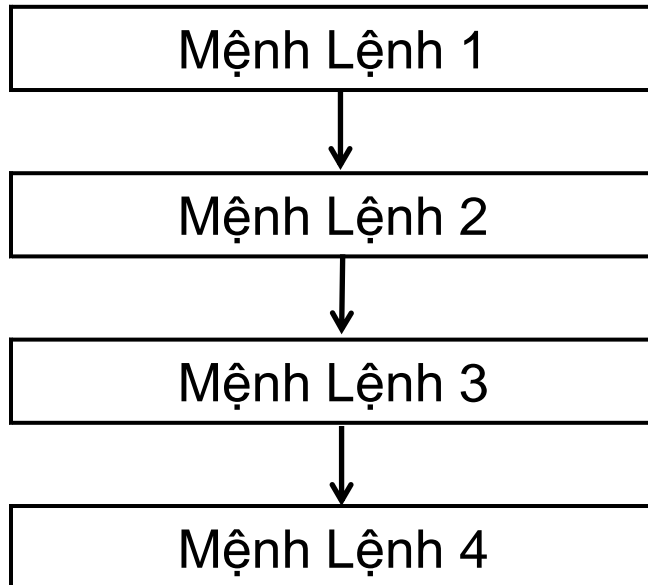
---

- ▶ Cấu trúc điều khiển
- ▶ Cấu trúc vòng lặp:
  - ▶ **while**
  - ▶ **do ... while**
  - ▶ **for**

# Cấu Trúc Điều Khiển – Tuần Tự

---

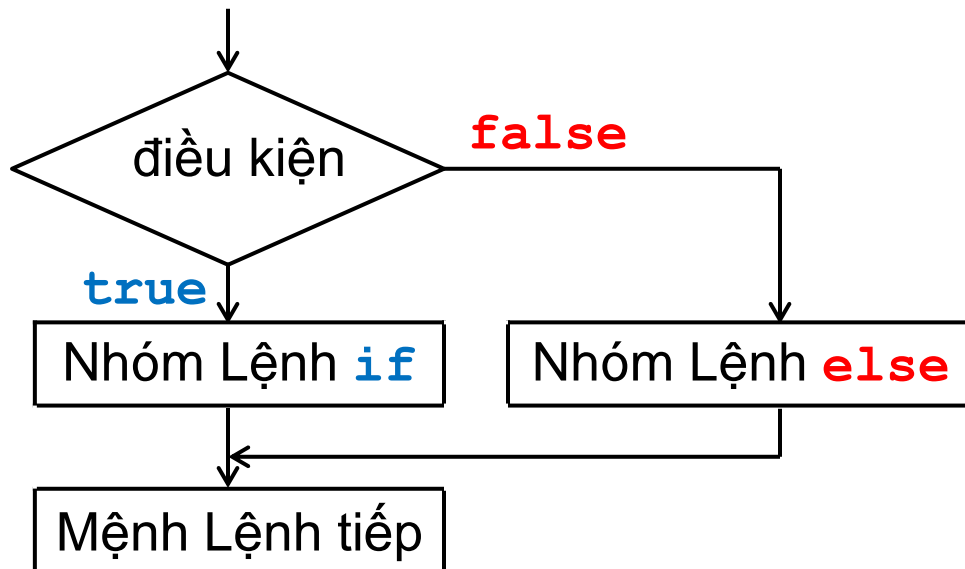
- ▶ Thứ tự tuần tự:
  - ▶ thực hiện mệnh lệnh theo thứ tự trong mã nguồn



```
cin >> m_mid;  
  
cin >> m_final;  
  
m_total = ... ..;  
  
cout << m_total;
```

# Cấu Trúc Điều Khiển – Lựa Chọn

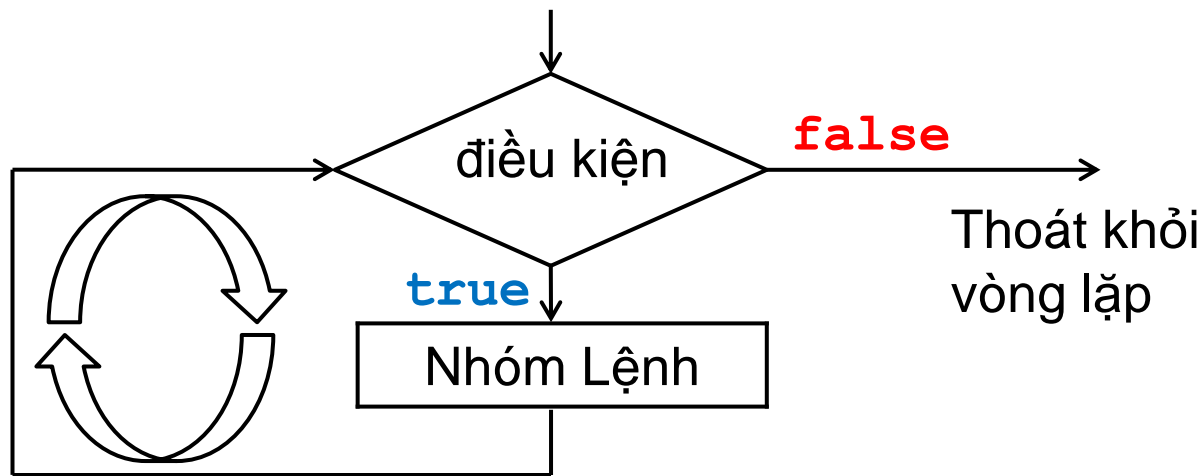
- ▶ Thứ tự lựa chọn:
  - ▶ Mệnh lệnh thực hiện phụ thuộc điều kiện



```
if (m_total < 4.0)
{
    cout << "truot";
}
else
{
    cout << "do";
}
... ..
```

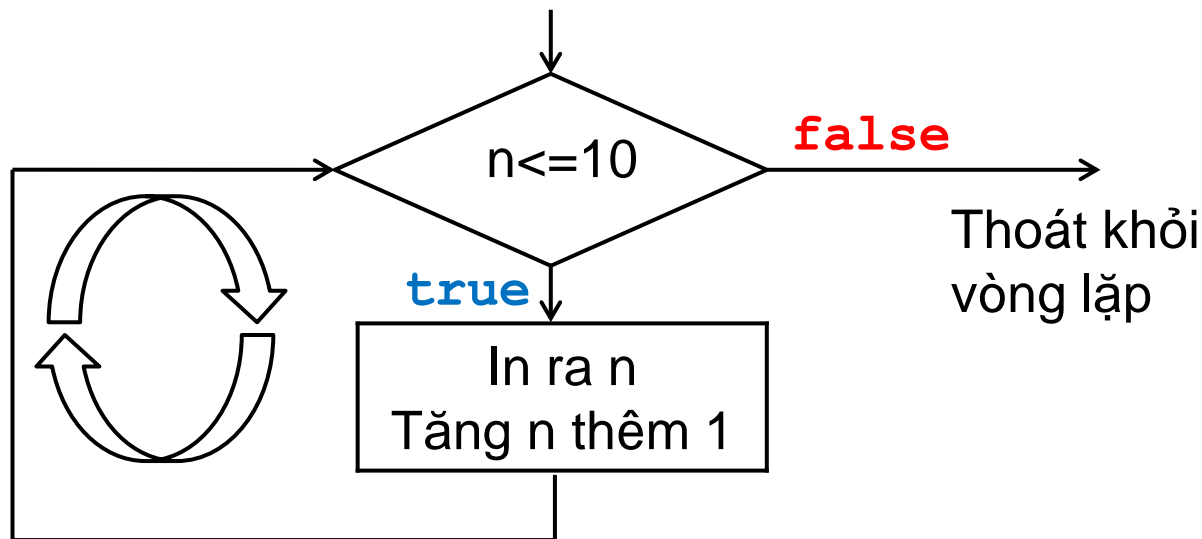
# Cấu Trúc Điều Khiển – Vòng Lặp

- ▶ Thứ tự vòng lặp:
  - ▶ Mệnh lệnh thực hiện lặp đi lặp lại phụ thuộc điều kiện



# Cấu Trúc Điều Khiển – Vòng Lặp

- ▶ Thứ tự vòng lặp:
  - ▶ Mệnh lệnh thực hiện lặp đi lặp lại phụ thuộc điều kiện
  - ▶ Ví dụ: in ra 10 số tự nhiên đầu tiên (đếm từ 1 đến 10)



# Cấu Trúc Điều Khiển – **while**

---

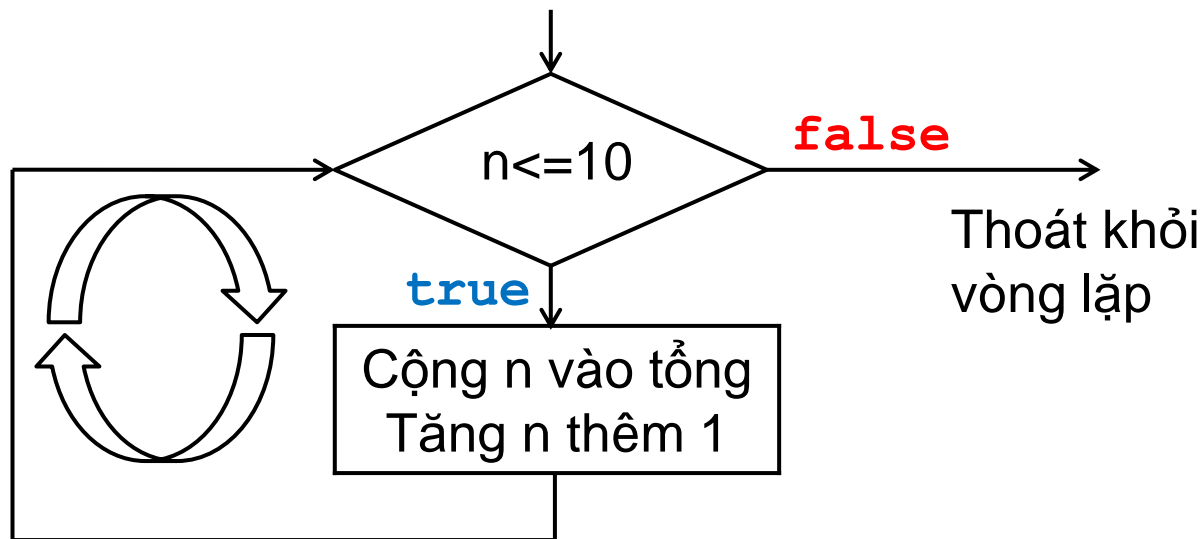
- ▶ Ví dụ: in ra 10 số tự nhiên đầu tiên

```
int n = 1;  // biến điều khiển vòng lặp

while (n <= 10)
{
    cout << n << endl;
    n = n + 1;
}
```

# Cấu Trúc Điều Khiển – **while**

- ▶ Thứ tự vòng lặp:
  - ▶ Mệnh lệnh thực hiện lặp đi lặp lại phụ thuộc điều kiện
  - ▶ Ví dụ: in ra tổng 10 số tự nhiên đầu tiên





# Cấu Trúc Điều Khiển – **while**

---

- ▶ Ví dụ: tính tổng 10 số tự nhiên đầu tiên

```
int sum = 0;

int n = 1;  // biến điều khiển vòng lặp

while (n <= 10)
{
    sum = sum + n;
    n = n + 1;
}

cout << sum << endl;
```

# Cấu Trúc Điều Khiển – **while**

---

- ▶ Ví dụ: tính tổng các số lẻ trong 10 số tự nhiên đầu tiên

```
int sum = 0;

int n = 1;  // biến điều khiển vòng lặp

while (n <= 10)
{
    if (n % 2 == 1) sum = sum + n;
    n = n + 1;
}

cout << sum << endl;
```

# Cấu Trúc Điều Khiển – **while**

---

- ▶ Ví dụ: tính tổng các số chẵn trong 10 số tự nhiên đầu tiên

```
int sum = 0;

int n = 2;  // biến điều khiển vòng lặp

while (n <= 10)
{
    sum = sum + n;
    n = n + 2;
}

cout << sum << endl;
```

# Cấu Trúc Điều Khiển – **while**

---

## ▶ Cú pháp:

```
while (biểu thức logic)
```

```
{
```

```
    // biểu thức logic đúng (true)
```

```
    // thực hiện mệnh lệnh trong thân cấu trúc while
```

```
    Mệnh Lệnh
```

```
}
```

```
... ..
```

## ▶ Ý nghĩa:

- ▶ nếu **biểu thức logic** đúng, thực hiện thân vòng lặp **while**
  - sau khi thực hiện thân vòng lặp **while**, quay lên kiểm tra **biểu thức logic**
- ▶ nếu **biểu thức logic** sai, thoát khỏi vòng lặp **while**

# Cấu Trúc Điều Khiển – **while**

- ▶ Ví dụ: In ra **N** dấu sao

Với **N = 4**

Giá trị của <b>i</b>	<b>i=0</b>	<b>i=1</b>	<b>i=2</b>	<b>i=3</b>	<b>i=4</b>
Điều khiển lặp	<b>true</b>	<b>true</b>	<b>true</b>	<b>true</b>	<b>false</b>
Kết quả in ra	*	*	*	*	

```
int i = 0; // biến đếm (biến điều khiển)

while (i < N)
{
    cout << "*";
    i = i + 1;
}
```

# Cấu Trúc Điều Khiển – **while**

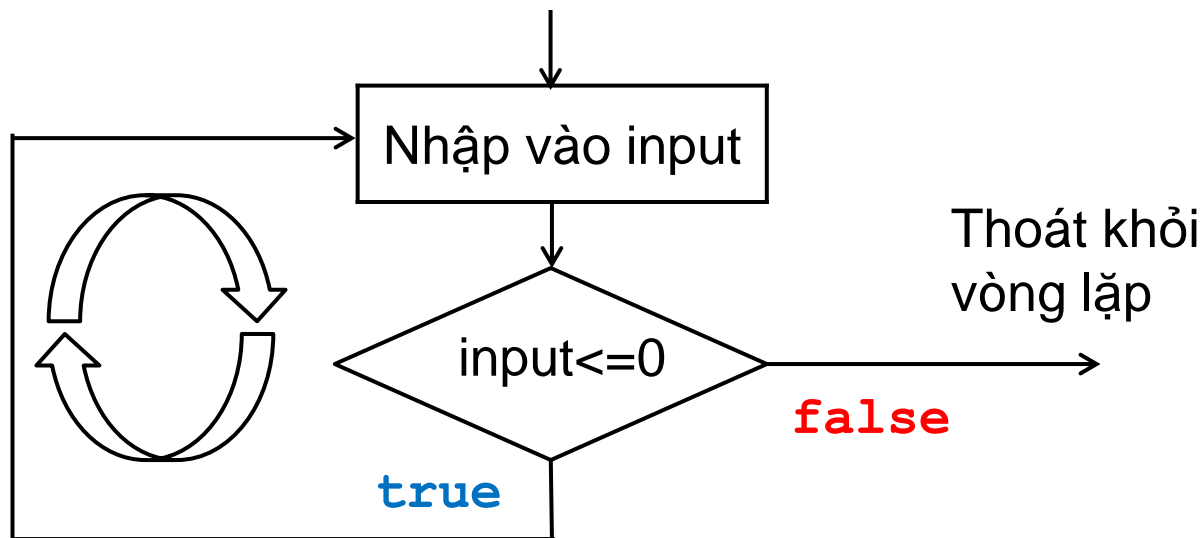
---

- ▶ Ví dụ: Yêu cầu nhập vào một số nguyên dương

```
int input;  
  
cin >> input;  
  
while (input <= 0)  
{  
    cin >> input;  
}
```

# Cấu Trúc Điều Khiển – **do...while**

- ▶ Thứ tự vòng lặp:
  - ▶ Mệnh lệnh thực hiện lặp đi lặp lại phụ thuộc điều kiện
  - ▶ Ví dụ: nhập vào một số nguyên dương



# Cấu Trúc Điều Khiển – **do...while**

---

- ▶ Ví dụ: Yêu cầu nhập vào một số nguyên dương

```
int input;  
  
do  
{  
    cin >> input;  
}  
while (input <= 0);
```



# Cấu Trúc Điều Khiển – **do...while**

---

## ▶ Cú pháp:

**do**

{

// biểu thức logic đúng (**true**)

// thực hiện mệnh lệnh trong thân cấu trúc do ... while

Mệnh Lệnh

} **while** (**biểu thức logic**) ;

... ..

## ▶ Ý nghĩa:

▶ thực hiện thân vòng lặp **do ... while**

▶ sau đó kiểm tra **biểu thức logic**

□ nếu đúng, thực hiện thân vòng lặp **do ... while**

□ nếu sai, thoát khỏi vòng lặp **do ... while**

# Cấu Trúc Điều Khiển-**while** / **do...while**

---

- ▶ Ví dụ: tính tổng các số nhập vào, kết thúc khi nhập vào số 0

Nhập số (0 để kết thúc) : 5

Tổng = 5

Nhập số (0 để kết thúc) : -2

Tổng = 3

Nhập số (0 để kết thúc) : 6

Tổng = 9

Nhập số (0 để kết thúc) : 0

# Cấu Trúc Điều Khiển–**while** / **do...while**

---

- ▶ Ví dụ: tính tổng các số nhập vào, kết thúc khi nhập vào số 0
- ▶ Khi nào thì lặp
  - Biểu thức điều kiện là gì
- ▶ Thân vòng lặp
  - Thực hiện những mệnh lệnh gì
- ▶ Cần bao nhiêu biến số
  - Ít nhất có biến điều khiển vòng lặp
- ▶ Khởi tạo biến điều khiển vòng lặp

# Cấu Trúc Điều Khiển-**while** / **do..while**

- ▶ Ví dụ: tính tổng các số nhập vào. Kết thúc khi nhập vào số 0

```
int sum = 0;

int input;

do
{
    cin >> input;
    sum = sum + input;
}
while (input != 0);

cout << sum << endl;
```

```
int sum = 0;

int input;
cin >> input;
while (input != 0)
{
    sum = sum + input;
    cin >> input;
}

cout << sum << endl;
```

# Cấu Trúc Điều Khiển-**while** / **do..while**

- ▶ Ví dụ: tính tổng các số nhập vào. Kết thúc khi nhập vào số nguyên âm

```
int sum = 0;

int input;

do
{
    cin >> input;
    sum = sum + input;
}
while (input >= 0);

cout << sum << endl;
```

```
int sum = 0;

int input;
cin >> input
while (input >= 0)
{
    sum = sum + input;
    cin >> input;
}

cout << sum << endl;
```

# Cấu Trúc Điều Khiển **while** / **do..while**

---

- ▶ Ví dụ: đoán số ngẫu nhiên tới khi đoán đúng

```
int so_lan_choi = 0;
int du_doan;
int ket_qua;
bool doan_dung;
do
{
    so_lan_choi = so_lan_choi + 1;
    cin >> du_doan;
    ket_qua = taoSoNgauNhien(XO_SO);
    doan_dung = ket_qua == du_doan;
}
while (!doan_dung);
cout << so_lan_choi << endl;
```

# Cấu Trúc Điều Khiển – **for**

---

## ▶ Cú pháp:

```
for (Mệnh Lệnh 1; biểu thức logic; Mệnh Lệnh 2)  
{  
    // biểu thức logic đúng (true)  
    // thực hiện mệnh lệnh trong thân cấu trúc for  
}
```

## ▶ Các bước thực hiện:

1. **Mệnh Lệnh 1** khởi tạo biến điều khiển
2. Nếu **biểu thức logic** đúng (**true**)
3. Thực hiện mệnh lệnh trong thân vòng lặp **for**
4. Thay đổi giá trị biến điều khiển (thực hiện **Mệnh Lệnh 2**)
5. Quay lại bước 2
6. Nếu sai, **biểu thức logic** sai (**false**), thoát khỏi vòng lặp

# Cấu Trúc Điều Khiển – **for**

- ▶ Ví dụ: In ra **N** dấu sao

Với **N = 4**

Giá trị của <b>i</b>	<b>i=0</b>	<b>i=1</b>	<b>i=2</b>	<b>i=3</b>	<b>i=4</b>
Điều khiển lặp	<b>true</b>	<b>true</b>	<b>true</b>	<b>true</b>	<b>false</b>
Kết quả in ra	*	*	*	*	

```
// i biến điều khiển (biến đếm)

for (int i = 0; i < N; i = i + 1)
{
    cout << "*";
}
```



# Cấu Trúc Điều Khiển – **while**

- ▶ Ví dụ: In ra **N** dấu sao

Với **N = 4**

Giá trị của <b>i</b>	<b>i=0</b>	<b>i=1</b>	<b>i=2</b>	<b>i=3</b>	<b>i=4</b>
Điều khiển lặp	<b>true</b>	<b>true</b>	<b>true</b>	<b>true</b>	<b>false</b>
Kết quả in ra	*	*	*	*	

```
// i biến điều khiển (biến đếm)
int i = 0;
while (          i < N          )
{
    cout << "*";
    i = i + 1;
}
```

# Cấu Trúc Điều Khiển – **while**

---

- ▶ Ví dụ: tính tổng 10 số tự nhiên đầu tiên

```
int sum = 0;

int n = 1;  // biến điều khiển vòng lặp

while (      n <= 10      )
{
    sum = sum + n;
    n = n + 1;
}

cout << sum << endl;
```

# Cấu Trúc Điều Khiển – **for**

---

- ▶ Ví dụ: tính tổng 10 số tự nhiên đầu tiên

```
int sum = 0;

        // biến điều khiển vòng lặp

for    (int n = 1; n <= 10; n = n + 1)
{
    sum = sum + n;

}

cout << sum << endl;
```

# Cấu Trúc Điều Khiển – **for**

---

- ▶ Ví dụ: tính tích 10 số tự nhiên đầu tiên

```
int product = 0;

        // biến điều khiển vòng lặp

for    (int n = 1; n <= 10; n = n + 1)
{
    product = product * n;

}

cout << product << endl;
```

# Cấu Trúc Vòng Lặp – So Sánh

---

- ▶ Ví dụ: Yêu cầu nhập vào một số nguyên dương

```
int input;  
  
do  
{  
    cin >> input;  
}  
while (input <= 0);
```

# Cấu Trúc Vòng Lặp – So Sánh

---

- ▶ Ví dụ: Yêu cầu nhập vào một số nguyên dương

```
int input;  
  
cin >> input;  
  
while (input <= 0)  
{  
    cin >> input;  
}
```

# Cấu Trúc Vòng Lặp – So Sánh

---

- ▶ Ví dụ: Yêu cầu nhập vào một số nguyên dương

```
int input;  
  
cin >> input;  
  
for (i input <= 0 i)  
{  
    cin >> input;  
}
```

# Cấu Trúc Vòng Lặp – So Sánh

---

- ▶ **while**

- ▶ Linh hoạt nhất
- ▶ Không có nhược điểm

- ▶ **do ... while**

- ▶ Kém linh hoạt
- ▶ Luôn thực hiện thân vòng lặp ít nhất một lần

- ▶ **for**

- ▶ Thường dùng khi biết chính xác số lần lặp



# Một Số Lưu Ý

---

- ▶ Vòng lặp vô hạn

- ▶ Cấu trúc **while**, có dấu “;”

```
while (input <= 0) ;  
{  
    cin >> input;  
}
```

- ▶ Sử dụng phép gán thay vì phép so sánh bằng

```
while (i = 1) {  
    cout << "*";  
}
```

- ▶ Khi **N** lẻ hoặc âm

```
int i = 0;  
while (i != N) {  
    i = i + 2;  
}
```

# break / continue

---

- ▶ Luồng điều khiển:
  - ▶ Cấu trúc lặp thể hiện một cách rõ ràng luồng điều khiển, khi nào bắt đầu, khi nào kết thúc một lần lặp
  - ▶ Trong một vài trường hợp, có thể cần thay đổi luồng lặp tự nhiên
- ▶ Mệnh lệnh **break**
  - ▶ Dừng vòng lặp ngay lập tức, kể cả khi điều kiện lặp vẫn đang đúng
- ▶ Mệnh lệnh **continue**
  - ▶ Bỏ qua phần còn lại của thân vòng lặp, bắt đầu một vòng lặp mới (kiểm tra điều kiện, lặp, ...)

# break / continue

---

- ▶ Ví dụ: nhập 4 số nguyên dương, và tính tổng

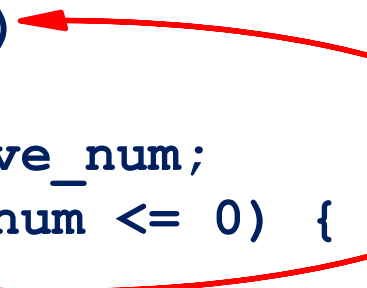
```
int positive_num, sum = 0;
int count = 1;
while (count <= 4)
{
    do
    {
        cin >> positive_num;
    } while (positive_num <= 0);
    sum = sum + positive_num;
    count = count + 1;
}
cout << sum << endl;
```

# break / continue

---

- ▶ Ví dụ: nhập 4 số nguyên dương, và tính tổng

```
int positive_num, sum = 0;
int count = 1;
while (count <= 4)
{
    cin >> positive_num;
    if (positive_num <= 0) {
        continue;
    }
    sum = sum + positive_num;
    count = count + 1;
}
cout << sum << endl;
```

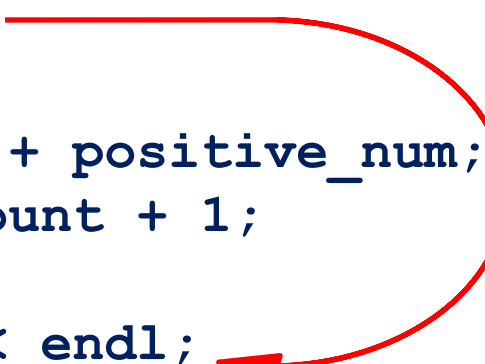


# break / continue

---

- ▶ Ví dụ: nhập 4 số nguyên dương, và tính tổng. Dừng khi nhập số âm

```
int positive_num, sum = 0;
int count = 1;
while (count <= 4)
{
    cin >> positive_num;
    if (positive_num <= 0) {
        break;
    }
    sum = sum + positive_num;
    count = count + 1;
}
cout << sum << endl;
```



## break / continue

---

- ▶ Ví dụ: nhập 4 số nguyên dương, và tính tổng. Dừng khi nhập số âm

```
int positive_num, sum = 0;
int count = 1;
while (count <= 4)
{
    cin >> positive_num;
    if (positive_num <= 0) count = 5;
    else {
        sum = sum + positive_num;
        count = count + 1;
    }
}
cout << sum << endl;
```