



# IT001 - NHẬP MÔN LẬP TRÌNH

## CHƯƠNG 4: CẤU TRÚC ĐIỀU KHIỂN FLOW CONTROL STRUCTURES (tt)

Khoa Khoa học Máy tính



# NỘI DUNG

- 4.1 Khái niệm câu lệnh và khối lệnh
- 4.2 Phạm vi hoạt động của biến
- 4.3 Giới thiệu về cấu trúc điều khiển
- 4.4 Cấu trúc rẽ nhánh if, if-else
- 4.5 Cấu trúc rẽ nhánh switch-case
- 4.6 Cấu trúc vòng lặp for
- 4.7 Cấu trúc vòng lặp while
- 4.8 Cấu trúc vòng lặp do-while
- 4.9 Lệnh break, continue, goto, return
- 4.10 Một số ví dụ minh họa
- Bài tập



## 4.6 Cấu trúc vòng lặp for



# Đặt vấn đề

- Viết chương trình xuất các số từ **1** đến **10**  
=> Sử dụng **10** câu lệnh cout
- Viết chương trình xuất các số từ **1** đến **1000**  
=> Sử dụng **1000** câu lệnh cout !

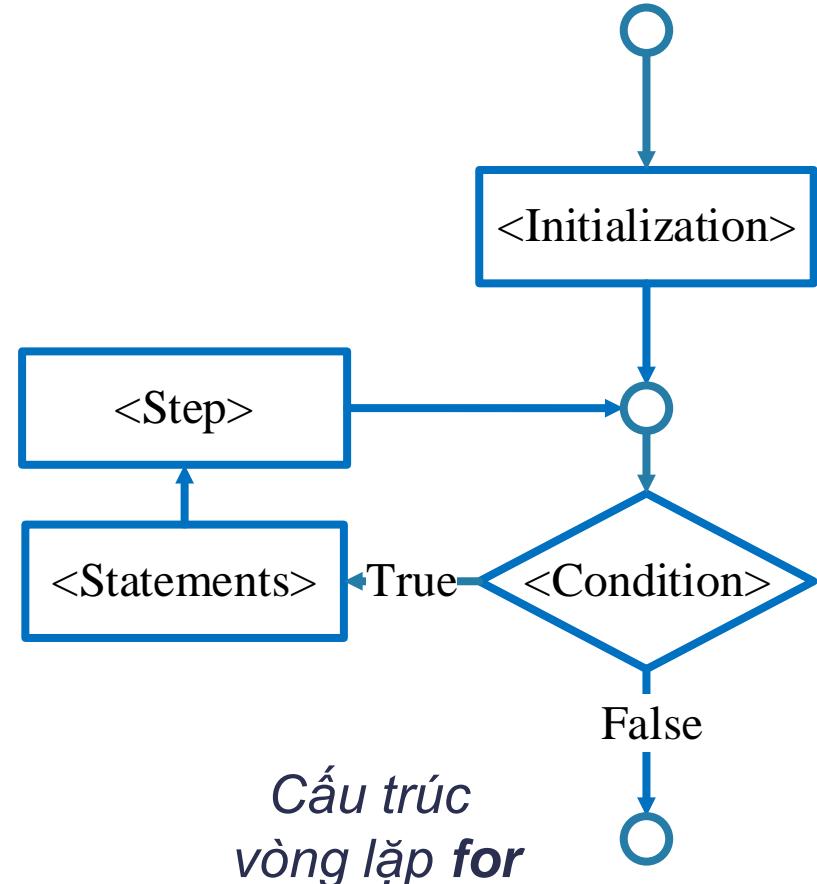
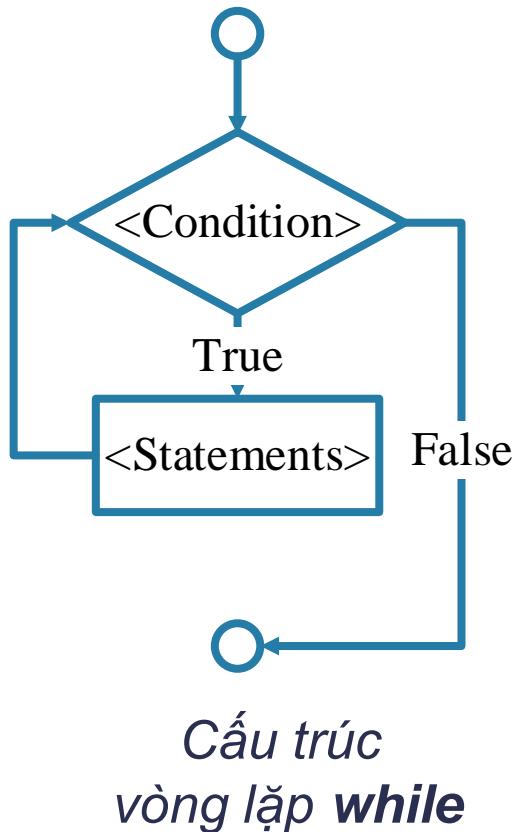
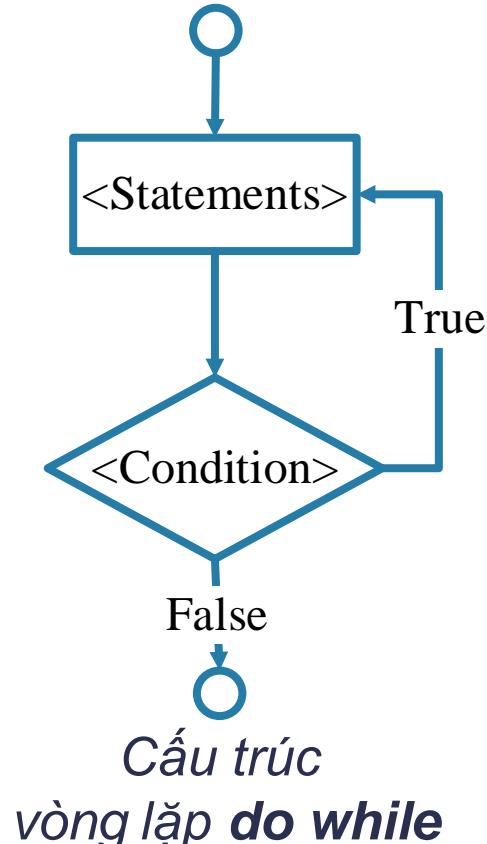
→ *Giải pháp:*

- Sử dụng **cấu trúc lặp** nhằm lập lại một hành động trong khi còn thỏa một điều kiện nào đó.
- 3 lệnh lặp: **for, while, do... while**



# Cấu trúc lặp - Iteration statements (Loops)

- Cấu trúc lặp: là việc lặp lại một câu lệnh một số lần nhất định hoặc thực hiện đến khi thỏa một điều kiện dừng, gồm: **while**, **do while** và **for**.





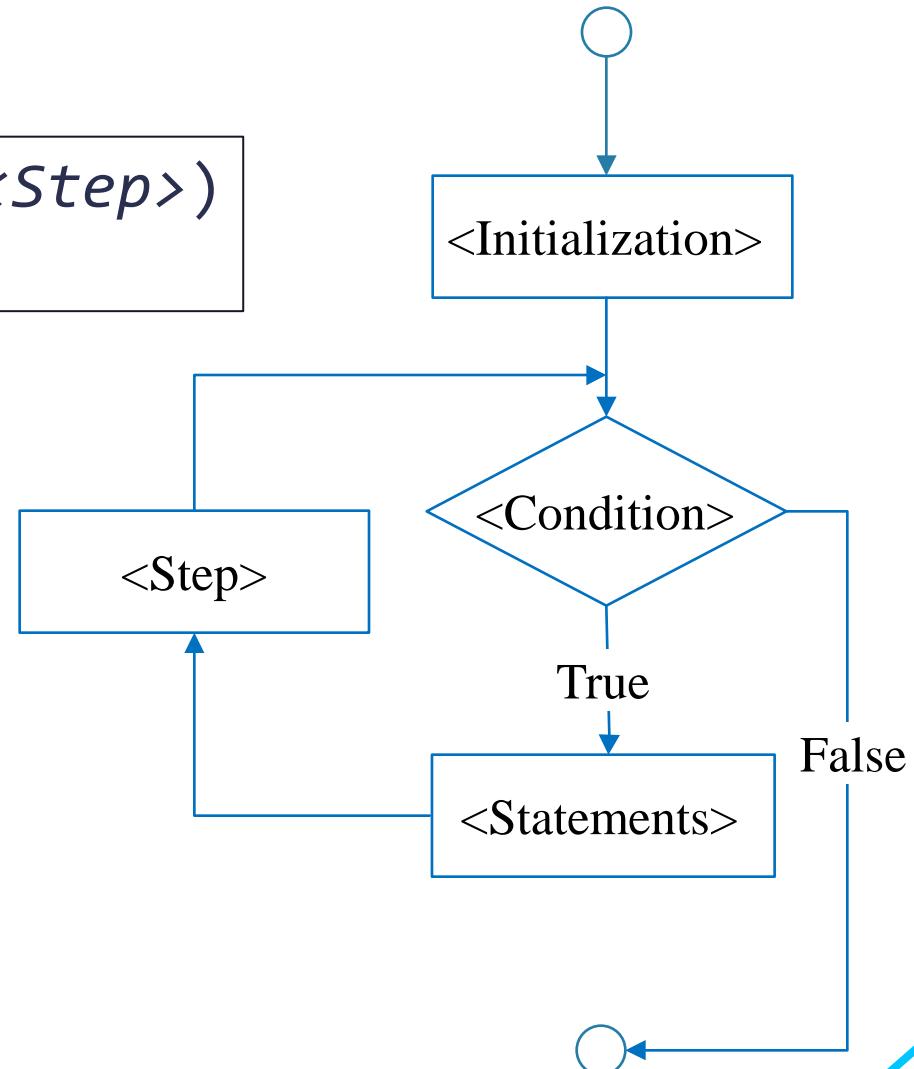
## 4.6 Cấu trúc vòng lặp for

- Cú pháp:

```
for (<Initialization> ; <Condition> ; <Step>)  
    <Statements>
```

Trong đó:

- Bước 1: Biểu thức khởi đầu *<Initialization>* được thực thi.
- Bước 2: Biểu thức điều kiện *<Condition>* được kiểm tra. Nếu đúng thì *<Statements>* được gọi, ngược lại thì kết thúc for.
- Bước 3: *<Step>* được thực thi, sau đó quay lại Bước 2





# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i){  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

i 0



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

i 0



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:  
i = 0

i 0



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:  
i = 0

i 0



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:  
i = 0

i

1



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:  
i = 0

i

1



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i

1



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i

1



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i

2



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i

2



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i = 2

i 2



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i = 2

i 2



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i = 2

i 3



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i = 2

i 3



# VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
  
}  
  
std::cout << "all done" << std::endl;
```

Kết quả xuất:

i = 0

i = 1

i = 2

all done

~~i 3~~



# Ví dụ:

```
int main() {  
    int i;  
    for(i=5; i<11; ++i)  
        cout << "i=" << i << endl;  
  
    cout << "Ngoai for i=" << i << endl;  
}
```



# Một số lưu ý for

- Câu lệnh **for** là một câu lệnh đơn và có thể lồng nhau.

```
if (n < 10 && m < 20) {  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < m; j++) {  
            cout << i + j;  
            cout << "\n";  
        }  
    }  
}
```



# Một số lưu ý for

- Trong câu lệnh for, có thể sẽ không có phần **initialization**.

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



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



# Một số lưu ý for

- Trong câu lệnh for, có thể sẽ không có phần **step**.

```
int i;
for (i = 0; i < 10; i++)
    cout << i << endl;
```



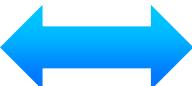
```
for (i = 0; i < 10; )
{
    cout << i << "\n";
    i++;
}
```



# Một số lưu ý for

- Trong câu lệnh for, có thể sẽ không có phần **condition**.
- Nếu không có **condition** thì câu lệnh for mặc định điều kiện luôn đúng => Cần dùng **break**, **return**, **continue**.

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



```
for (i = 0; ; i++) {
    if (i >= 10) break;
    cout << i << "\n";
}
```

```
for (i = 0; ; i++)
    cout << i << "\n";
```

? Điều kiện dừng



# Một số lưu ý for

- Không được thêm dấu chấm phẩy ; ngay sau lệnh lệnh for.  
=> Tương đương câu lệnh rỗng.

```
for (i = 0; i < 10; i++); {  
    cout << i;  
    cout << "\n";  
}
```



```
for (i = 0; i < 10; i++) {  
};  
{  
    cout << i;  
    cout << "\n";  
}
```



# Một số lưu ý for

- Các thành phần **initialization, condition, step** cách nhau bằng dấu chấm phẩy ;
- Nếu có nhiều thành phần trong mỗi phần thì được cách nhau bằng dấu phẩy ,

```
for (int i = 1, j = 2; i + j < 10; i++, j += 2)  
    cout << i + j << endl;
```



# Ví dụ: Tính giai thừa của n: $n!=1*2*3*..*n$

```
int n;
```

```
cin >> n; // n là số dương
```

```
long long S=1;
```

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

```
    S *= i;
```

```
cout << S;
```



# Ví dụ: Tính tổng giai thừa: $1!+2!+3!+\dots+n!$

```
int n;  
cin >> n; // n là số dương  
long long temp=1;  
long long S=0;  
  
for(int i = 1; i <= n; i++) {  
    temp *= i;  
    S += temp;  
}  
cout << S;
```



## 4.7 Cấu trúc vòng lặp while



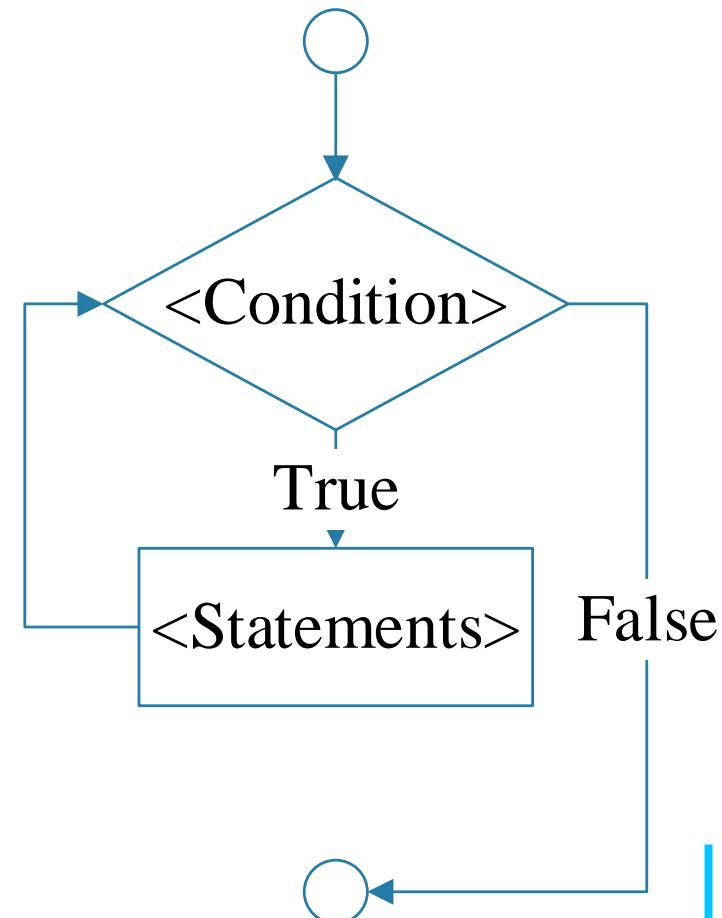
## 4.7 Cấu trúc vòng lặp while

- Cú pháp:

```
while (<Condition>)
    <Statements>
```

- **Cách bước thực hiện:**

- Bước 1: Biểu thức điều kiện *<Condition>* được kiểm tra. Nếu đúng thì chuyển sang Bước 2, ngược lại thì kết thúc while.
- Bước 2: Thực thi các lệnh *<Statements>*. Sau đó quay lại Bước 1.
- Lưu ý: *<Statements>* có thể là một câu lệnh hoặc một khối lệnh (khối lệnh thì phải đặt trong dấu khối lệnh)





# Ví dụ minh họa:

Chạy từng bước đoạn code sau:

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```



# Chạy từng bước

```
int n = 3;  
  
int count = 0;  
  
double sum = 0;  
  
while (count < n) {  
  
    double value;  
  
    std::cin >> value;  
  
    sum += value;  
  
    count++;  
  
}  
  
double average = sum / count ;  
  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n 3



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	0



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	0
sum	0



# Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	0
sum	0



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	0
sum	0
value	



# Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	0
sum	0
value	1



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	0
sum	1
value	1



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	1
sum	1
value	1



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	1
sum	1



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	1
sum	1
value	



# Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, **5**, 3, 1 }

n	3
count	1
sum	1
value	5



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	1
sum	6
value	5



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	2
sum	6
value	5



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	2
sum	6



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	2
sum	6
value	



# Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, **3**, 1 }

n	3
count	2
sum	6
value	3



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	2
sum	9
value	3



# Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++; // Line highlighted in yellow
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	3
sum	9
value	3



# Chạy từng bước

```
int n = 3;                                Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

count=3 < n=3 => Sai

n	3
count	3
sum	9



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	3
sum	9
average	3



# Chạy từng bước

```
int n = 3;  
int count = 0;  
double sum = 0;  
while (count < n) {  
    double value;  
    std::cin >> value;  
    sum += value;  
    count++;  
}  
double average = sum / count ;  
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào theo thứ tự: { 1, 5, 3, 1 }

n	3
count	3
sum	9
average	3



# Một số lưu ý

- Câu lệnh while có thể không thực hiện lần nào do điều kiện lặp ngay từ lần đầu đã không thỏa.

```
int n = 1;  
while (n > 10) {  
    cout << n << endl;  
    n--;  
}
```



# Một số lưu ý

- Cấu trúc for có thể được viết lại sử dụng cấu trúc while như sau:

```
for (initialization; condition; step)  
    statements
```



```
initialization;  
while (condition) {  
    statements  
    step;  
}
```



# Một số lưu ý

- Không được thêm ; ngay sau lệnh lệnh while.

```
int n = 0;  
while (n < 10);  
{  
    cout << n << "\n";  
    n++;  
}
```



```
int n = 0;  
while (n < 10){  
};  
{  
    cout << n << "\n";  
    n++;  
}
```



# Một số lưu ý

- Câu lệnh **while** có thể bị lặp vô tận (**infinite loop**).
- Hỏi kết quả của 2 đoạn code sau:

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

```
int n = 1;
while (n < 10)
    cout << n << endl;
```



# Ví dụ: Tính giai thừa của n: $n!=1*2*3*..*n$

```
int n;  
cin >> n; // n là số dương  
long long S=1;  
  
for(int i = 1; i <= n; i++)  
    S *= i;  
  
cout << S;
```

```
int n;  
cin >> n; // n là số dương  
long long S=1;  
  
int i=1;  
while(i<=n) {  
    S *= i;  
    i++;  
}  
cout << S;
```



## 4.8 Cấu trúc vòng lặp do-while



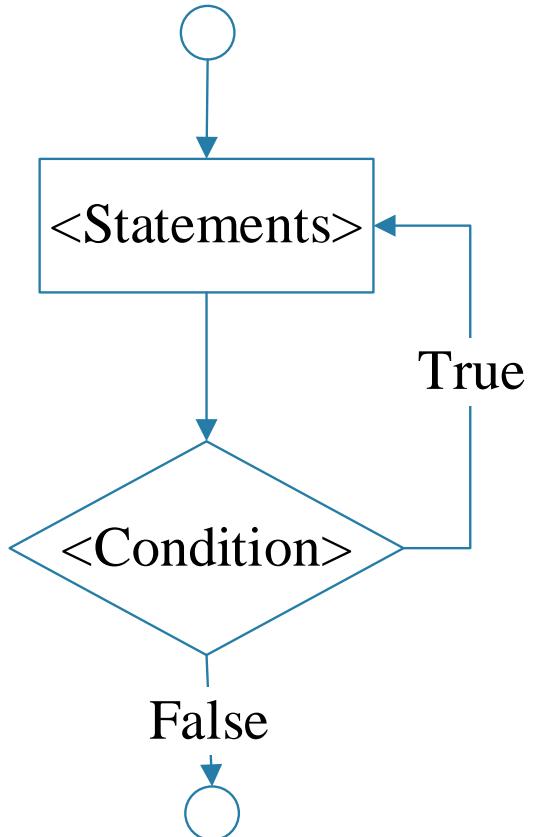
## 4.8 Cấu trúc vòng lặp while

- Cú pháp:

```
do  
    <Statements>  
  while (<Condition>);
```

- Các bước thực hiện:

- Bước 1: Thực thi các lệnh *<Statements>*
- Bước 2: Biểu thức điều kiện *<Condition>* được kiểm tra. Nếu đúng thì quay lại Bước 1, ngược lại thì kết thúc do-while.
- Lưu ý: *<Statements>* có thể là một câu lệnh hoặc một khối lệnh (khối lệnh thì phải đặt trong dấu khối lệnh)





# Một số lưu ý

- Câu lệnh **do... while** là một câu lệnh đơn và có thể lồng nhau.

```
int a = 1, b;  
do {  
    b = 1;  
    do {  
        cout << "%d\n", a + b);  
        b = b + 2;  
    } while (b < 20);  
    a++;  
} while (a < 20);
```



# Một số lưu ý

- Câu lệnh do... while sẽ được thực hiện ít nhất 1 lần do điều kiện lặp được kiểm tra ở cuối.

```
#include<iostream>
int main() {
    int n;
    do {
        std::cout << "Nhập n: ";
        std::cin >> n;
    } while (n < 1 || n > 100);
}
```



# Một số lưu ý

- Câu lệnh **do... while** có thể bị lặp vô tận (**infinite loop**)

```
int n = 1;
do {
    cout << n;
    n--;
} while (n < 10);
```



```
int n = 1;
do
    cout << n;
while (n < 10);
```



# Vòng lặp cho các khoảng giá trị - Range-based for loop

- Cú pháp: **for** ( declaration : range ) statement;
- Declaration: khai báo biến để lấy giá trị từng phần tử của range
- Range chứa các phần tử nối tiếp nhau: chuỗi, mảng, container hoặc kiểu dữ liệu hỗ trợ hàm bắt đầu và kết thúc .



# Ví dụ:

```
// range-based for loop  
#include <iostream>  
#include <string>  
using namespace std;  
  
int main () {  
    string str {"Hello!"};  
    for (char c : str) {  
        cout << "[" << c << "]";  
    }  
    cout << '\n';  
}
```



## 4.9 Câu lệnh break, continue, goto, return

---

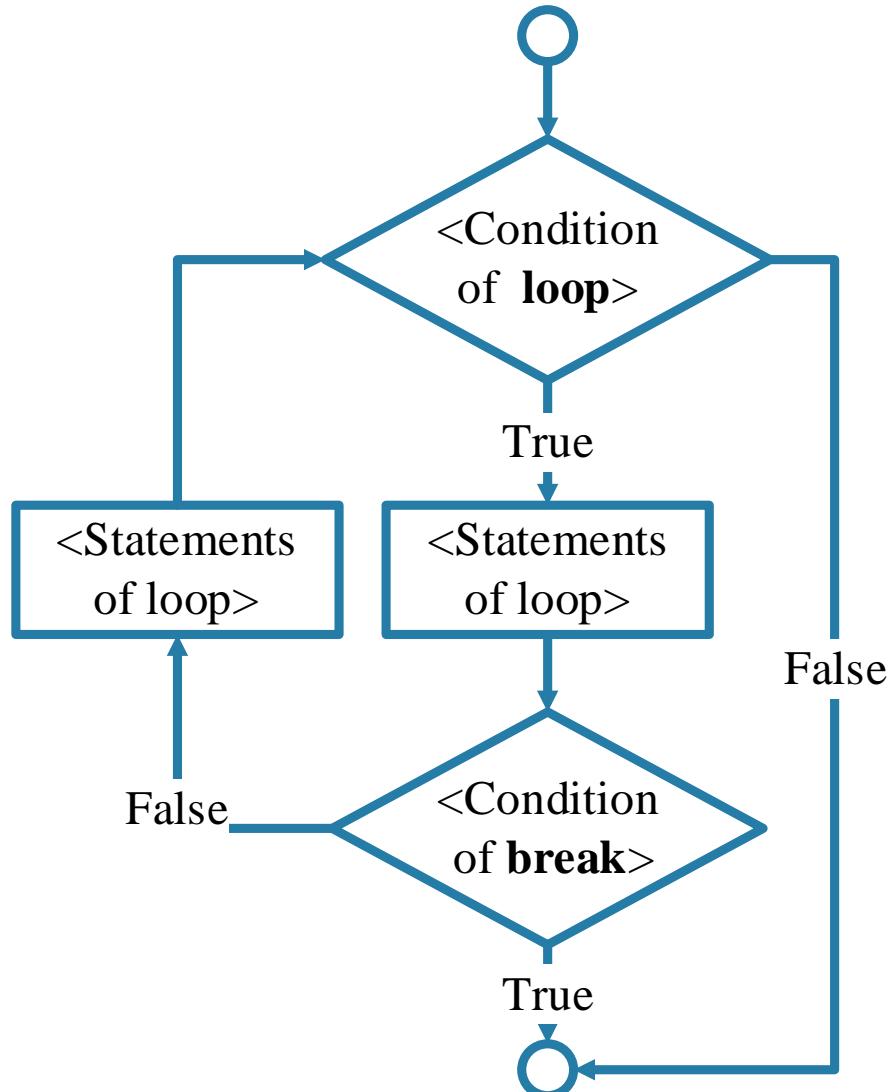


# Lệnh break

- Lệnh **break**:
  - Cho phép ra khỏi lệnh for, while, do while, switch nếu thỏa điều kiện.
  - Chương trình sẽ tiếp tục thực hiện các câu lệnh tiếp sau câu lệnh vừa thoát.

- Ví dụ:

```
for (i = 0; i < 10; i++) {  
    if (i % 2 == 0)  
        break;  
    cout << i;  
}
```



Lưu đồ lệnh lặp kết hợp lệnh **break**



# Ví dụ: break with Nested loop

```
// using break statement inside
// nested for loop
#include <iostream>
using namespace std;
int main() {
    int number;
    int sum = 0;
    // nested for loops
    // first loop
    for (int i = 1; i <= 3; i++) {
        // second loop
        for (int j = 1; j <= 3; j++) {
            if (i == 2) break;
            cout << "i = " << i << ", j = " << j << endl;
        }
    }
    return 0;
}
```

Kết quả:

```
i = 1, j = 1
i = 1, j = 2
i = 1, j = 3
i = 3, j = 1
i = 3, j = 2
i = 3, j = 3
```



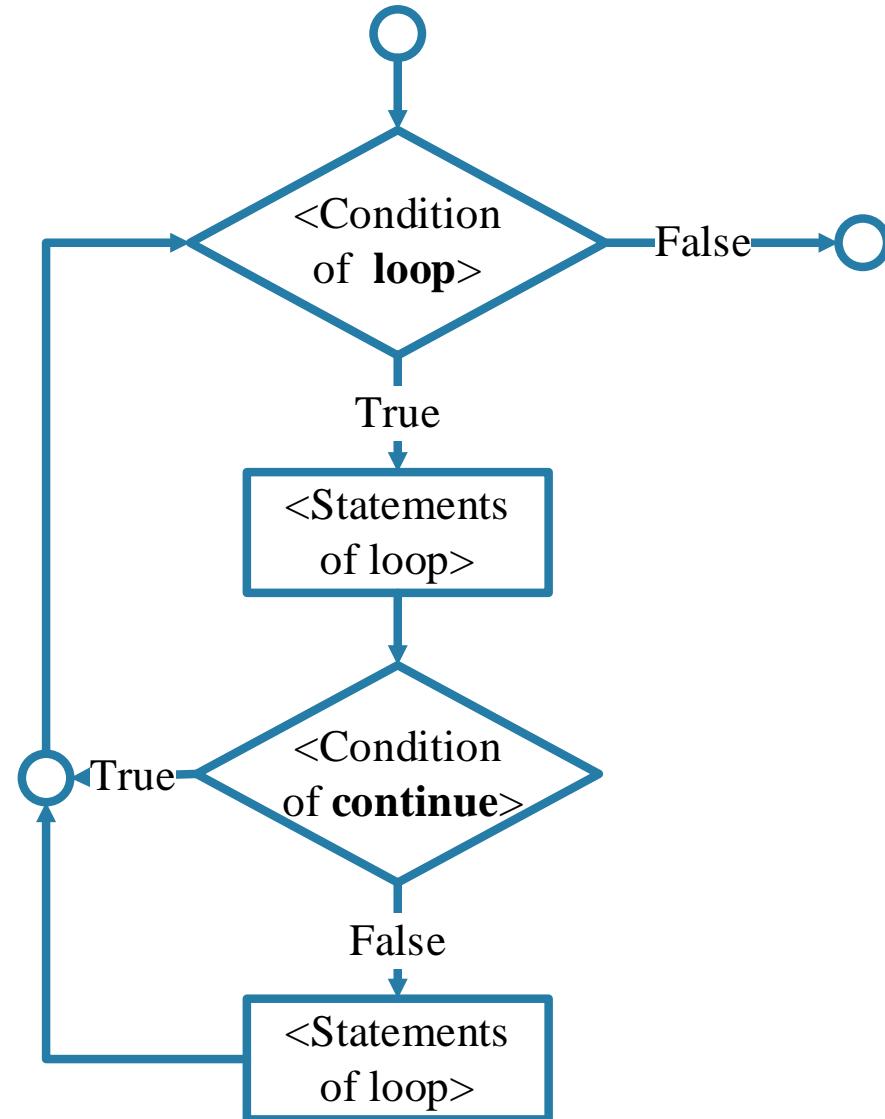
# Lệnh continue

- Lệnh **continue** :

- Lệnh dùng để quay lại đầu vòng lặp mà không chờ thực hiện hết các lệnh trong khối lệnh lặp.
  - Lệnh for: chuyển đến xét step tiếp theo.
  - while/do while: chuyển tới xác định biểu thức điều kiện và kiểm tra điều kiện kết thúc chương trình.

Ví dụ:

```
for (i = 0; i < 10; i++) {  
    if (i % 2 == 0) continue;  
    cout << i;  
}
```



Lưu đồ lệnh **lặp** kết hợp lệnh **continue**



# Lệnh continue

Cho biết kết quả của đoạn chương trình sau:

```
#include <iostream>
using namespace std;
int main() {
    int i = 1;
    do {
        cout << i;
        i++;
        if (i < 15)
            continue;
        cout << "in";
    } while (false);
    return 0;
}
```

Lưu ý: Vòng lặp do while kiểm tra điều kiện sau **mỗi** lần lặp. Vì vậy, sau câu lệnh continue, điều khiển sẽ chuyển sang câu lệnh while(false).



# Lệnh goto

- **goto** cho phép chương trình chuyển đến thực hiện một đoạn lệnh khác bắt đầu từ một điểm được đánh dấu bởi một **label (nhãn)** trong chương trình.
- **Label** là một tên gọi do người lập trình tự đặt theo các quy tắc đặt tên gọi.
- Lệnh goto thường được sử dụng để tạo vòng lặp.
- Tuy nhiên việc xuất hiện nhiều lệnh goto dẫn đến việc khó theo dõi trình tự thực hiện chương trình, vì vậy lệnh này thường được sử dụng rất hạn chế.

- Ví dụ:

```
#include <iostream>
using namespace std;
int main () {
    int n=10;
    mylabel:
    cout << n << " ";
    n--;
    if (n>=0) goto mylabel;
    cout << "end.\n";
}
```

Kết quả: 10 9 8 7 6 5 4 3 2 1 0 end.



# Lệnh goto

```
#include <iostream>
using namespace std;

int main() {
    int control = 5;
    if (control == 1) goto LABEL_1;
    else if (control == 2) goto LABEL_2;
    else if (control == 3) goto LABEL_3;
    else if (control == 4) goto LABEL_4;
    else if (control == 5) goto LABEL_5;
    else goto DEFAULT;
}

DEFAULT: cout << "go to DEFAULT" << endl;
        cout << "Hi!" << endl;
        cout << "How are you?" << endl;
LABEL_5: cout << "go to LABEL_5" << endl;
LABEL_4: cout << "go to LABEL_4" << endl;
LABEL_3: cout << "go to LABEL_3" << endl;
LABEL_2: cout << "go to LABEL_2" << endl;
LABEL_1: cout << "go to LABEL_1" << endl;
}
```

Kết quả:  
go to LABEL\_5  
go to LABEL\_4  
go to LABEL\_3  
go to LABEL\_2  
go to LABEL\_1

```
#include <iostream>
using namespace std;

int main() {
    int control = 5;
    switch (control) {
        default: cout << "go to DEFAULT" << endl;
                  cout << "Hi!" << endl;
                  cout << "How are you?" << endl;
        case 5: cout << "go to LABEL_5" << endl;
        case 4: cout << "go to LABEL_4" << endl;
        case 3: cout << "go to LABEL_3" << endl;
        case 2: cout << "go to LABEL_2" << endl;
        case 1: cout << "go to LABEL_1" << endl;
    }
}
```

Kết quả:  
go to LABEL\_5  
go to LABEL\_4  
go to LABEL\_3  
go to LABEL\_2  
go to LABEL\_1



# Lệnh return

- Khi gặp lệnh **return** máy sẽ kết thúc hàm chứa nó.
- Tùy theo giá trị trả về của hàm mà return có kèm giá trị hay không.

- Ví dụ:

```
#include <iostream>
using namespace std;
```

```
int main() {
    int pass, count=1;
    while(1) {
        cin >> pass;
        if (count == 5) return 0;
        // Nhập sai lần thứ 5, thoát khỏi chương trình
```

```
        if (pass != 123456) {
            ++count;
            continue;
        } // Nhập sai mật khẩu, tiếp tục nhập
```

```
        break; // Nhập đúng, kết thúc vòng lặp
    }
```

```
    cout << "Welcome to the example program";
    return 0;
}
```



## 4.10 Một số ví dụ minh họa



## 4.10 Một số ví dụ minh họa

- Ví dụ 1: Viết chương trình Nhập một số nguyên dương n (có kiểm tra điều kiện nhập) và tính tổng  $S=1+2+\dots+n$
- Ví dụ 2: Viết chương trình Liệt kê tất cả các ước số của số nguyên dương n
- Ví dụ 3: Viết chương trình Đếm số lượng chữ số của số nguyên dương n
- Ví dụ 4: Viết chương trình Kiểm tra số nguyên tố (có dùng break)
- Ví dụ 5: Viết chương trình In tất cả các số lẻ nhỏ hơn 50 trừ các số 3,9,31 (có dùng continue)



# Bài tập



# Bài tập

1. Viết chương trình nhập vào số nguyên dương n. Tính tổng:

$$S = \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{2n}$$

2. Viết chương trình nhập vào số nguyên dương n. Tính tổng:  $S = 1 + 1.2 + \dots + 1.2.3\dots.n.$

3. Viết chương trình liệt kê tất cả các số nguyên tố nhỏ hơn giá trị N nhập từ bàn phím ( $N < 100$ ).

4. Viết chương trình tính tổng các chữ số trong 1 số Ví dụ: số 1234 có tổng  $S = 1 + 2 + 3 + 4 = 10$ .

5. Tìm ước số chung lớn nhất của 2 số nguyên dương a và b.



# Chúc các em học tốt !

