

LECTURE 03

LOOP STATEMENT



pythonTM

Big-O Coding

Website: www.bigocoding.com

Giới thiệu

BT1 – TÍNH TỔNG

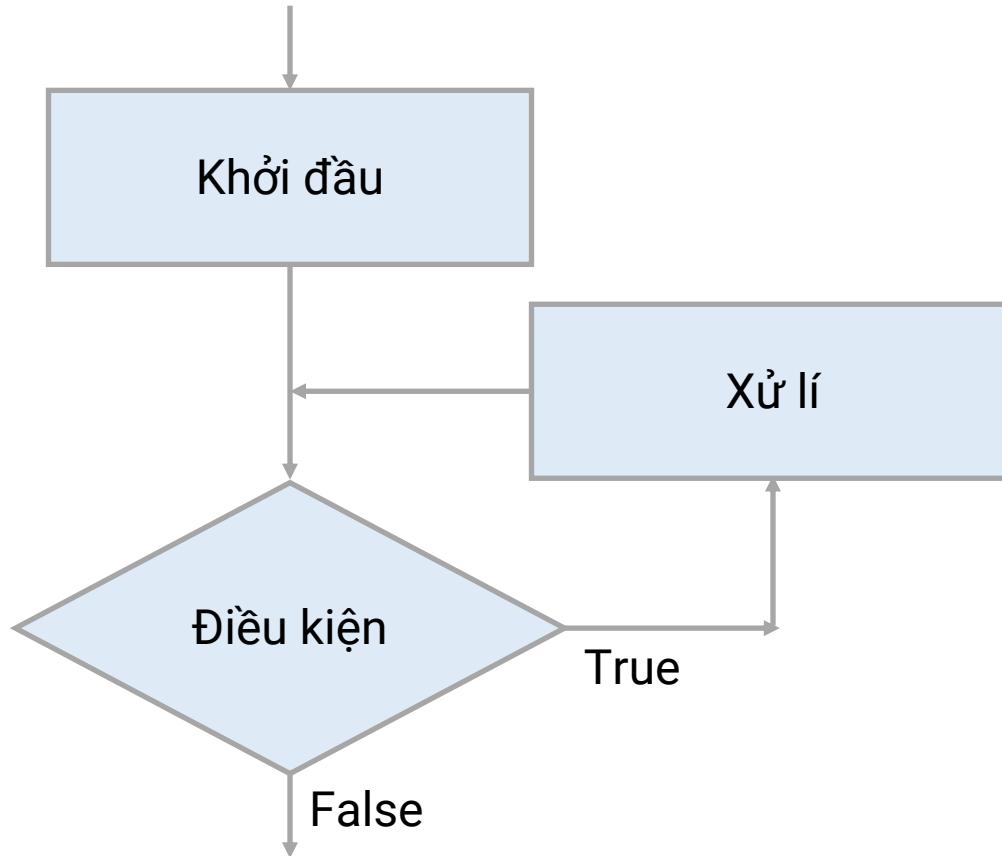
- Viết chương trình đọc vào số nguyên dương n và tính tổng các số nguyên từ 1 đến n.
- Ví dụ:
 - Input: 5
 - Output: 15 ($1 + 2 + 3 + 4 + 5 = 15$)



BT1 – Giải thuật

1. Sử dụng lệnh nhập, đọc vào số nguyên dương n.
2. Bắt đầu từ 0: ans = 0.
3. **Lần lượt** cộng với 1, 2, 3, 4,... đến khi gấp n thì dừng. Ví dụ: n = 5:
 1. $\text{ans} = \text{ans} + 1 = 0 + 1 = 1$
 2. $\text{ans} = \text{ans} + 2 = 1 + 2 = 3$
 3. $\text{ans} = \text{ans} + 3 = 3 + 3 = 6$
 4. $\text{ans} = \text{ans} + 4 = 6 + 4 = 10$
 5. $\text{ans} = \text{ans} + 5 = 10 + 5 = 15 \rightarrow \text{Đừng.}$
4. Sử dụng lệnh xuất, in kết quả ra màn hình.

Sơ đồ của câu lệnh lặp

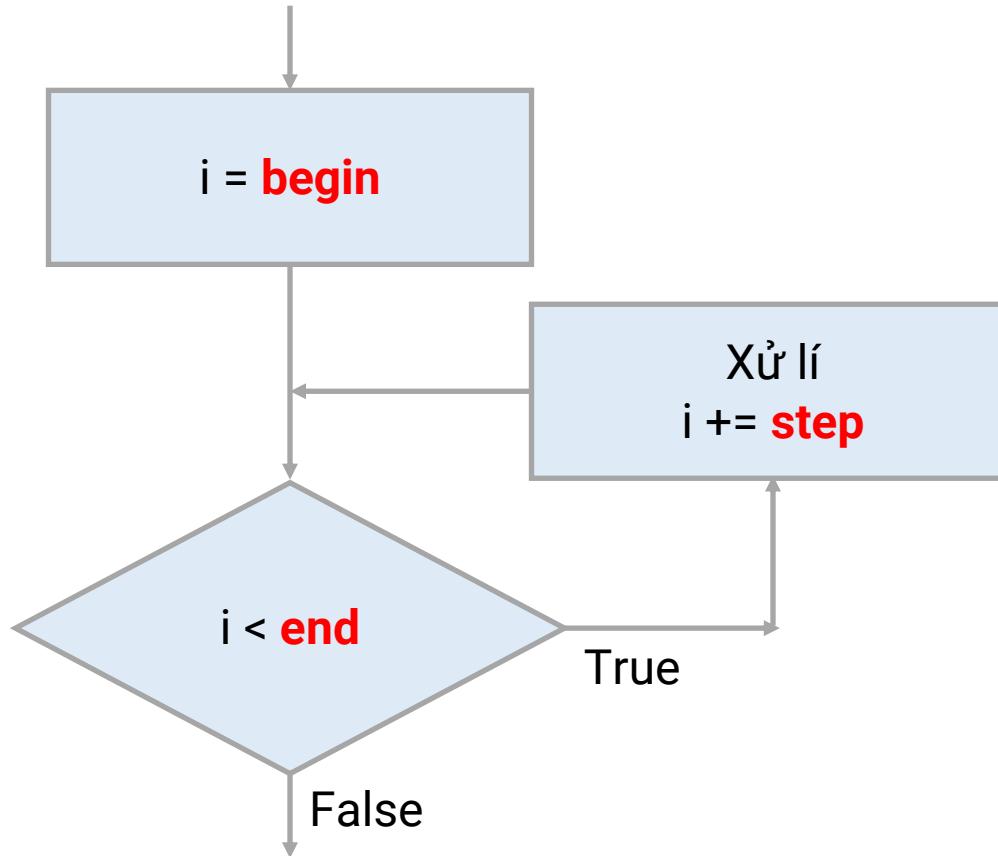


Câu lệnh lặp

- C++, Java: có 3 câu lệnh lặp: for, while và do...while.
- Python: có 2 câu lệnh lặp: for, while.

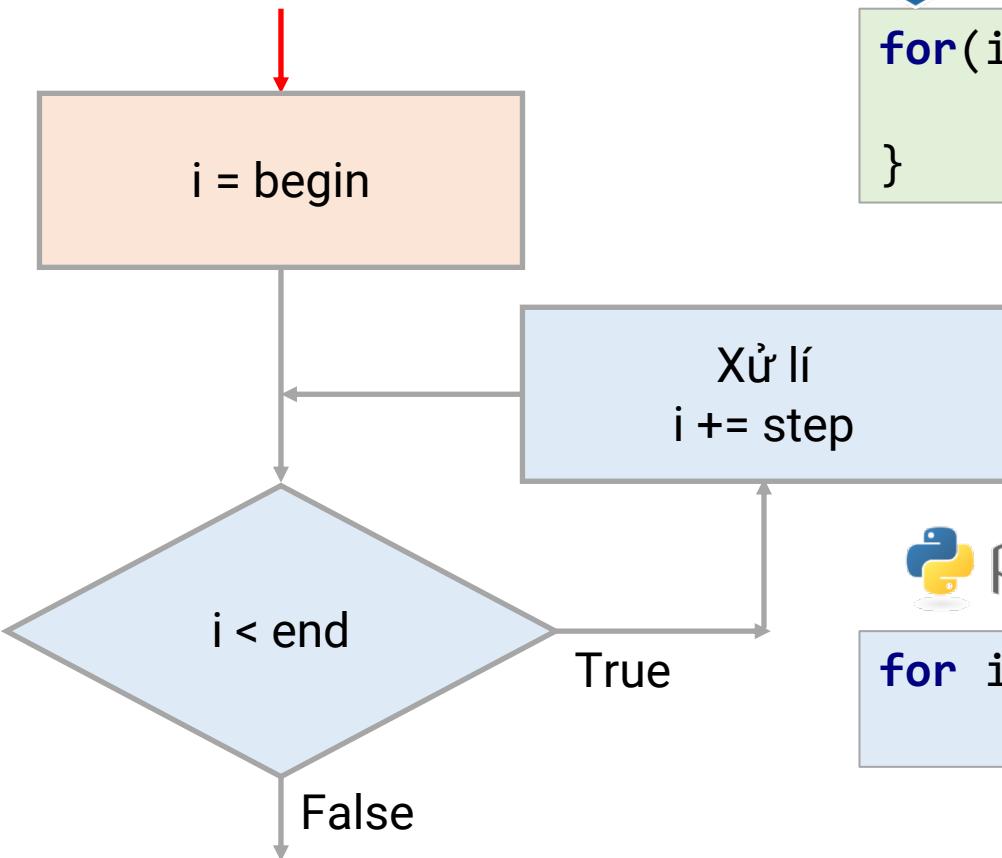
Câu lệnh for

Câu lệnh for



Khi xác định đầy đủ: begin, end, step

Câu lệnh for

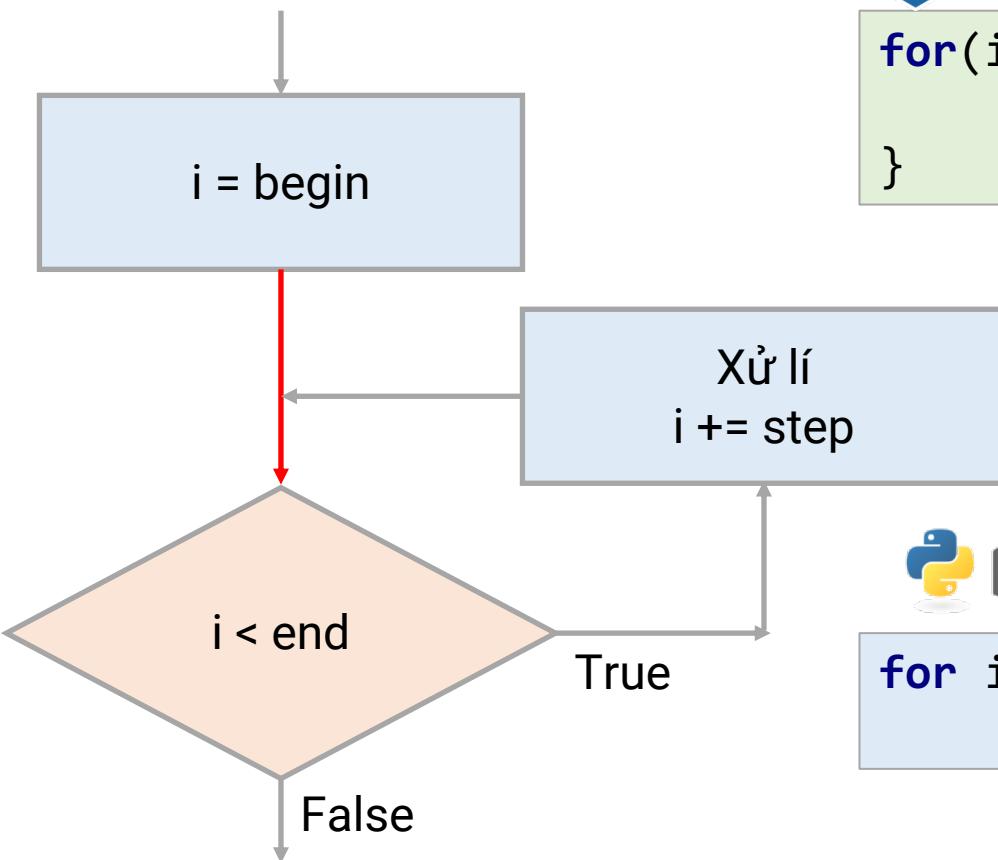


```
for(i = begin; ;){  
}
```



```
for i in range(begin, , ):
```

Câu lệnh for

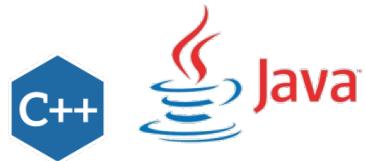
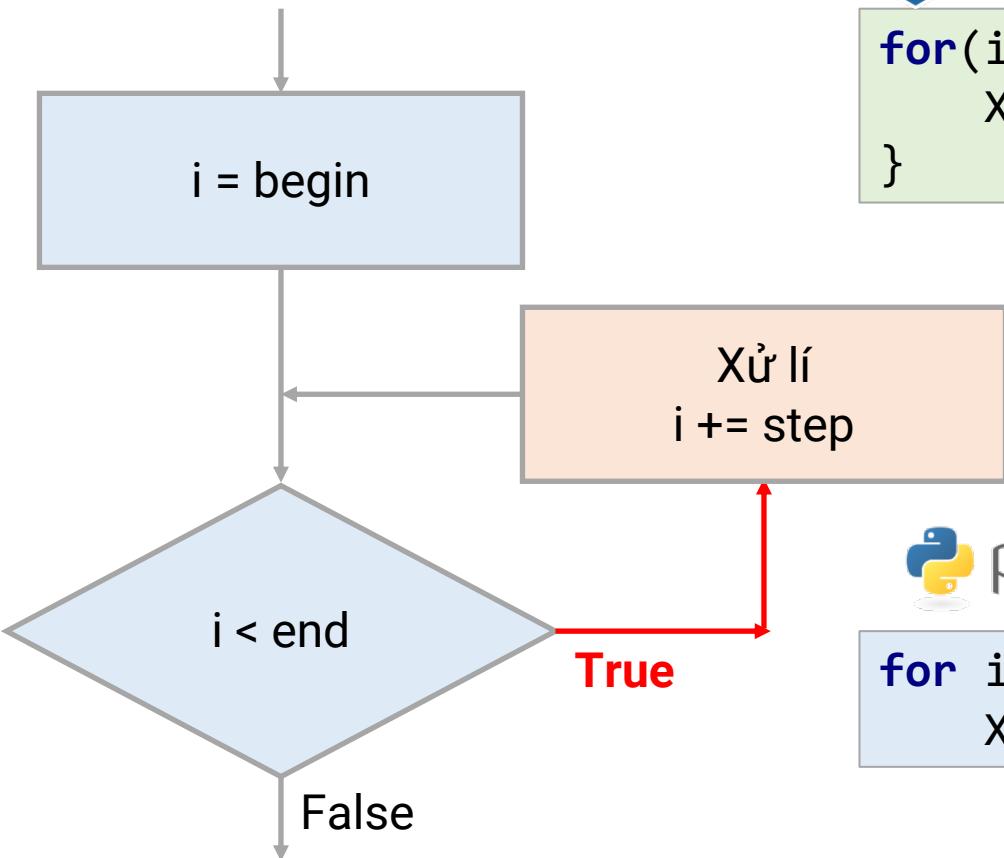


```
for(i = begin; i < end; ){  
}
```



```
for i in range(begin, end, ):
```

Câu lệnh for

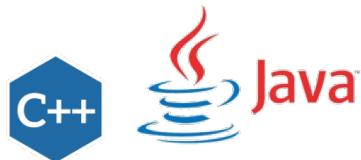
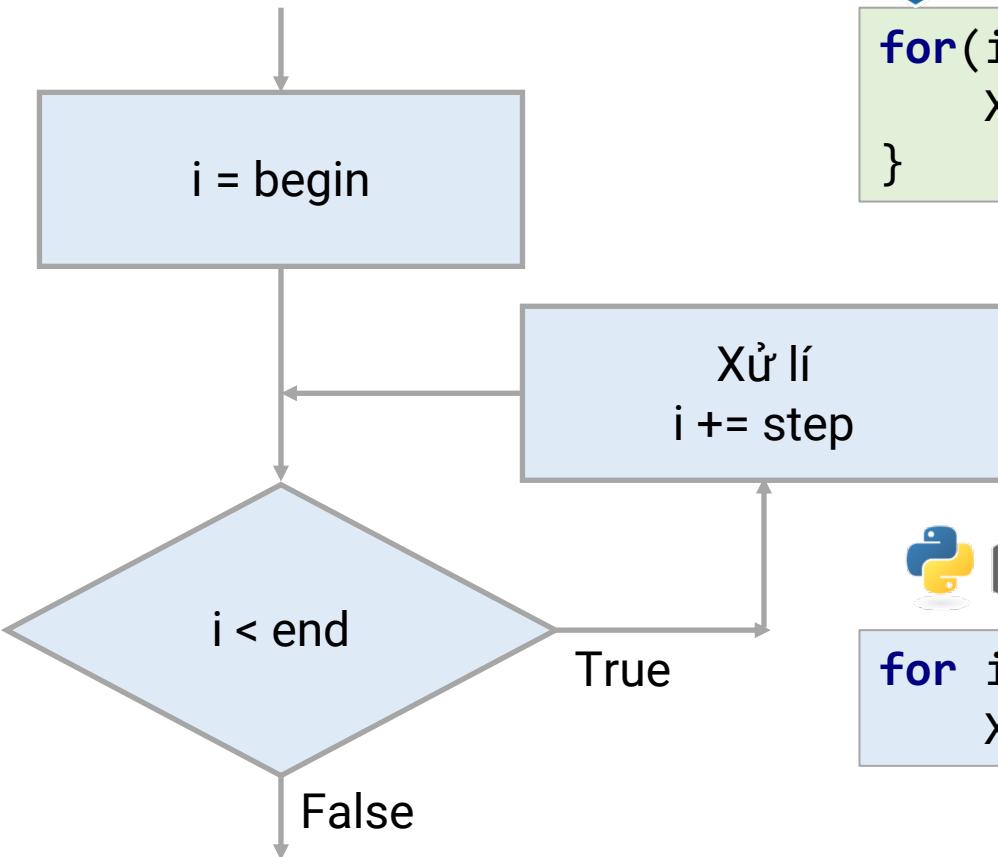


```
for(i = begin; i < end; i += step){  
    Xử lí;  
}
```



```
for i in range(begin, end, step):  
    Xử lí
```

Câu lệnh for



```
for(i = begin; i < end; i += step){
    Xử lí;
}
```

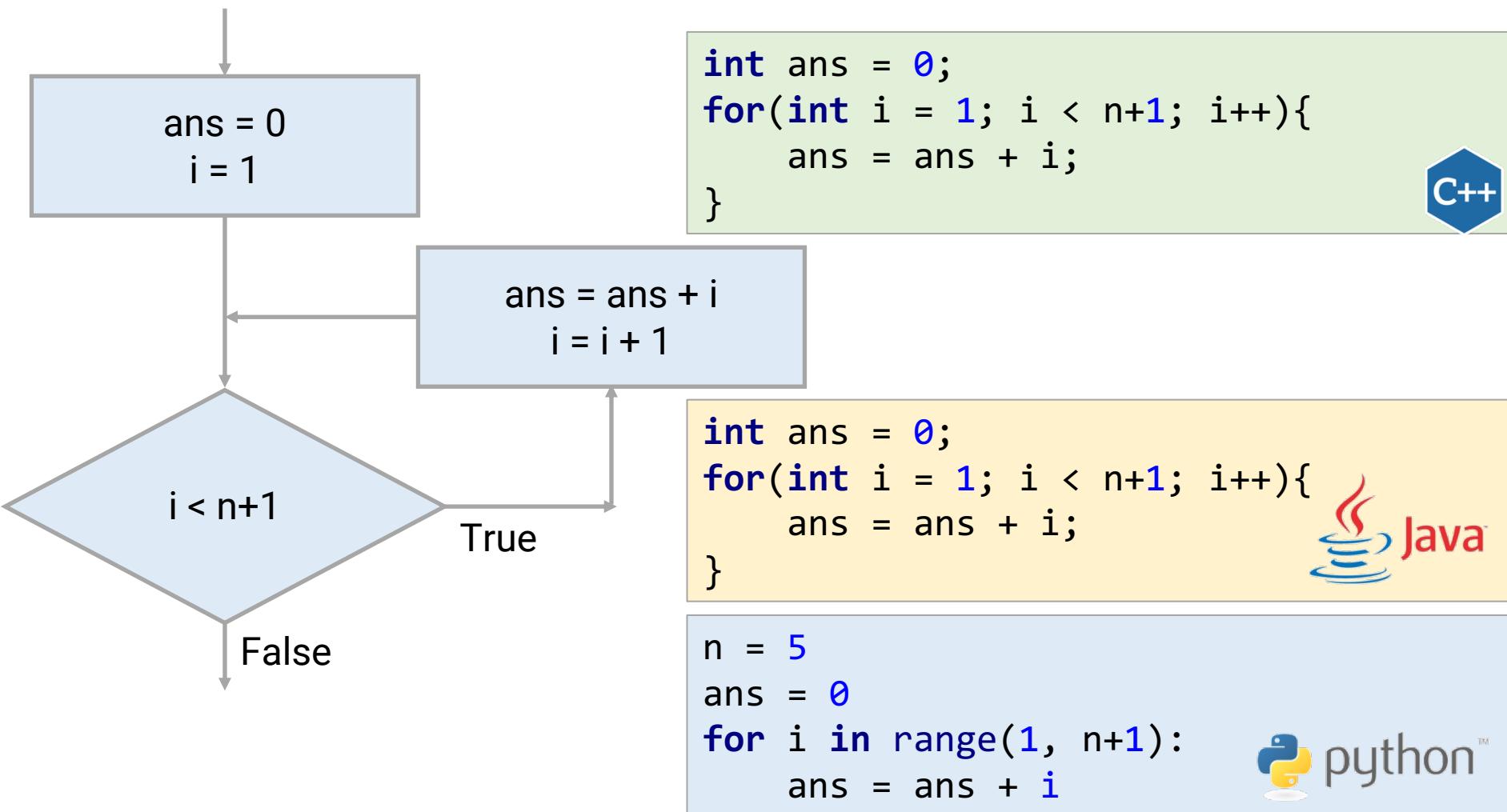
- Cặp dấu {}
- Không có ; ở dòng for
- Không bắt buộc viết lùi "Xử lí" vào 1 tab



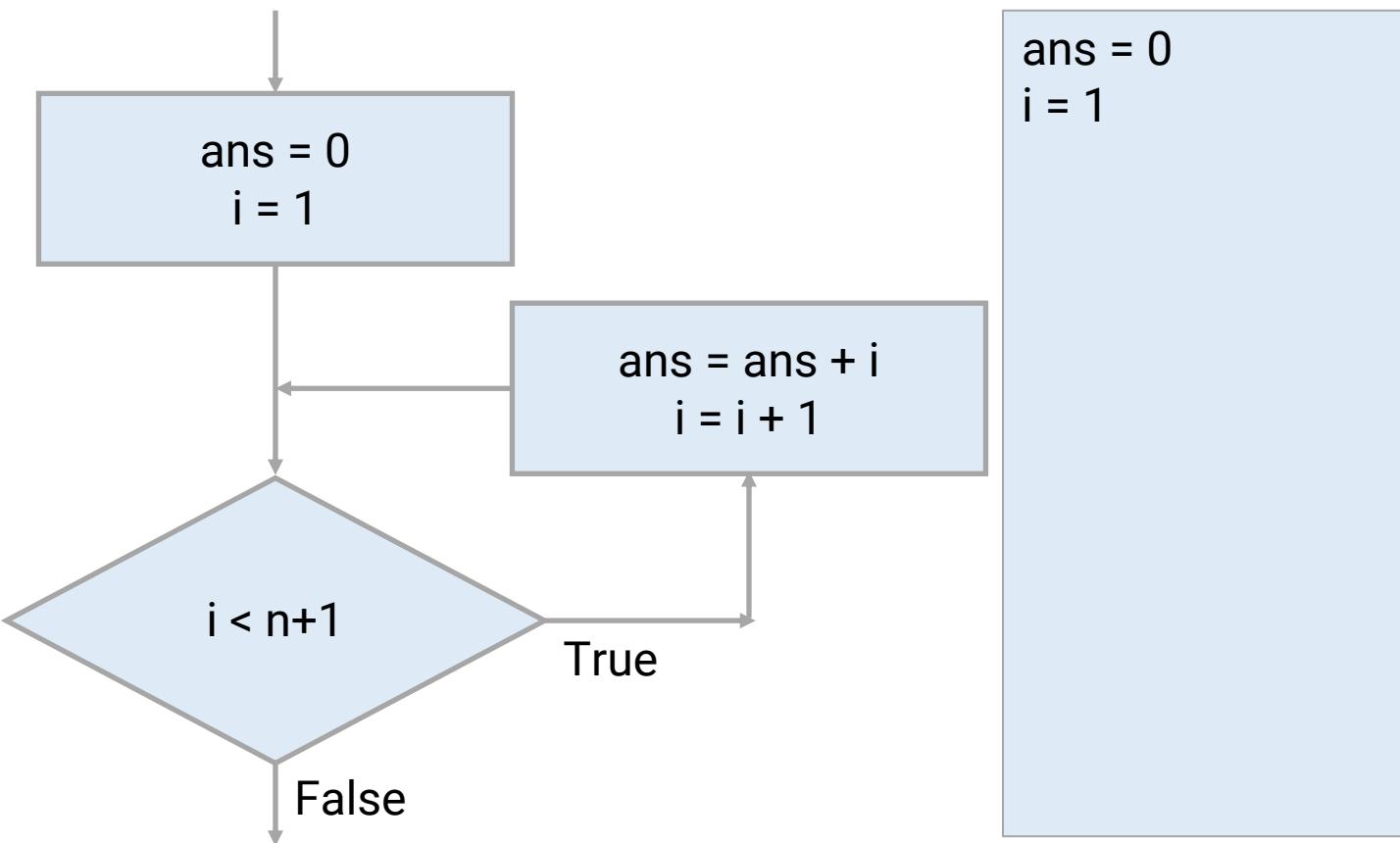
```
for i in range(begin, end, step):
    Xử lí
```

- Không cần cặp dấu {}
- Phải có dấu :
- Bắt buộc viết lùi "Xử lí" vào 1 tab

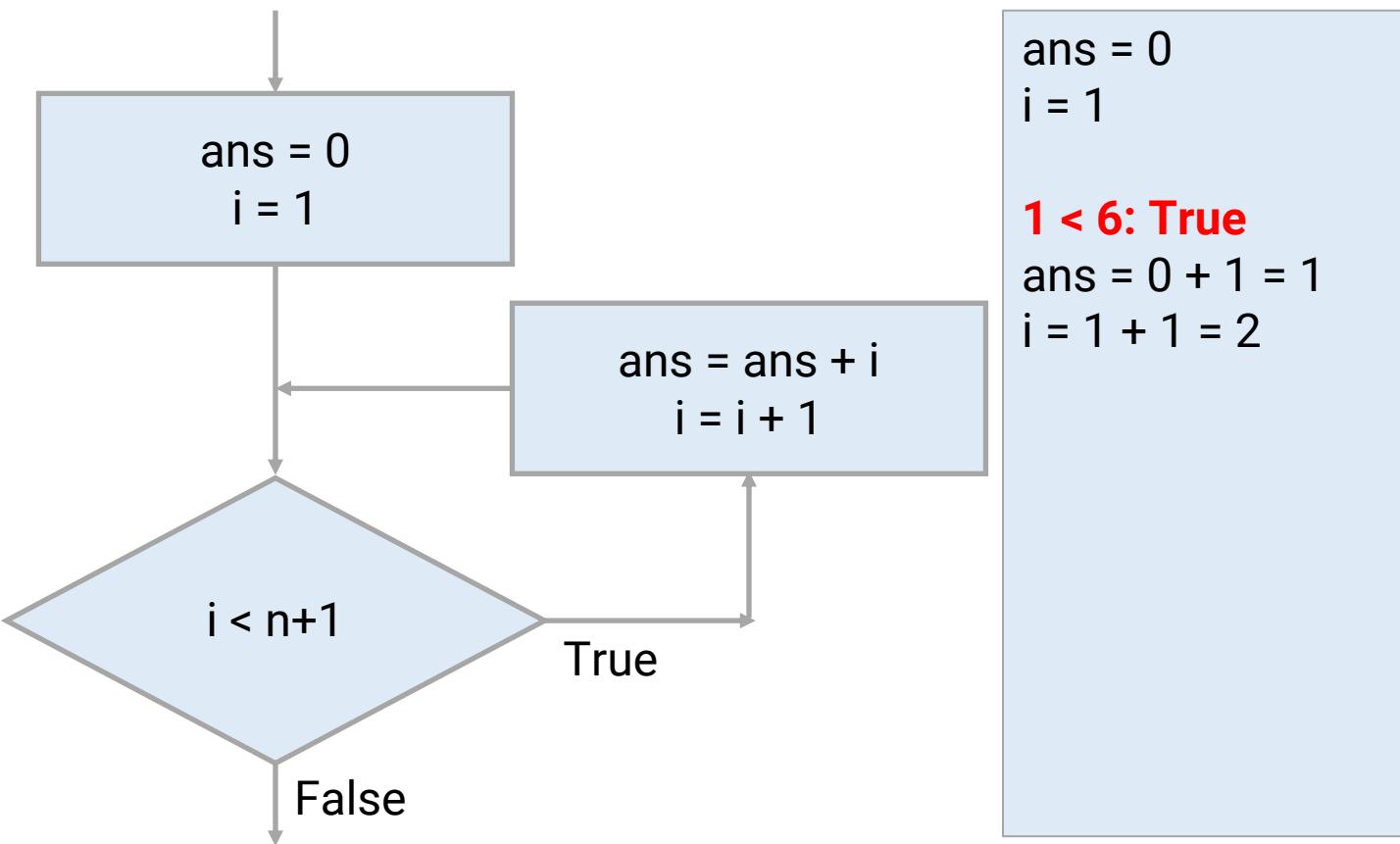
VD: Tính tổng các số từ 1 đến n



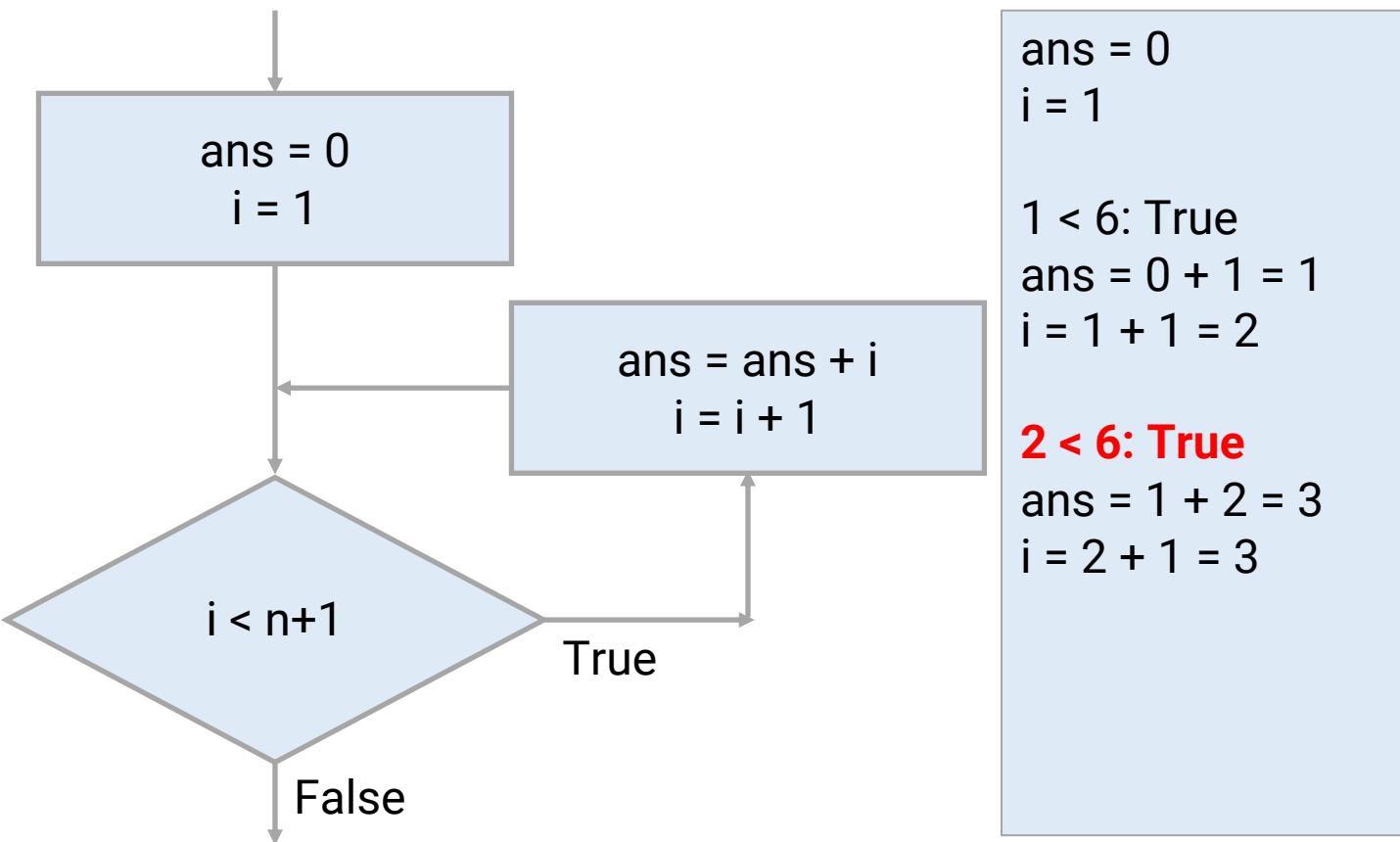
VD: Tính tổng các số từ 1 đến n = 5 – Step by step



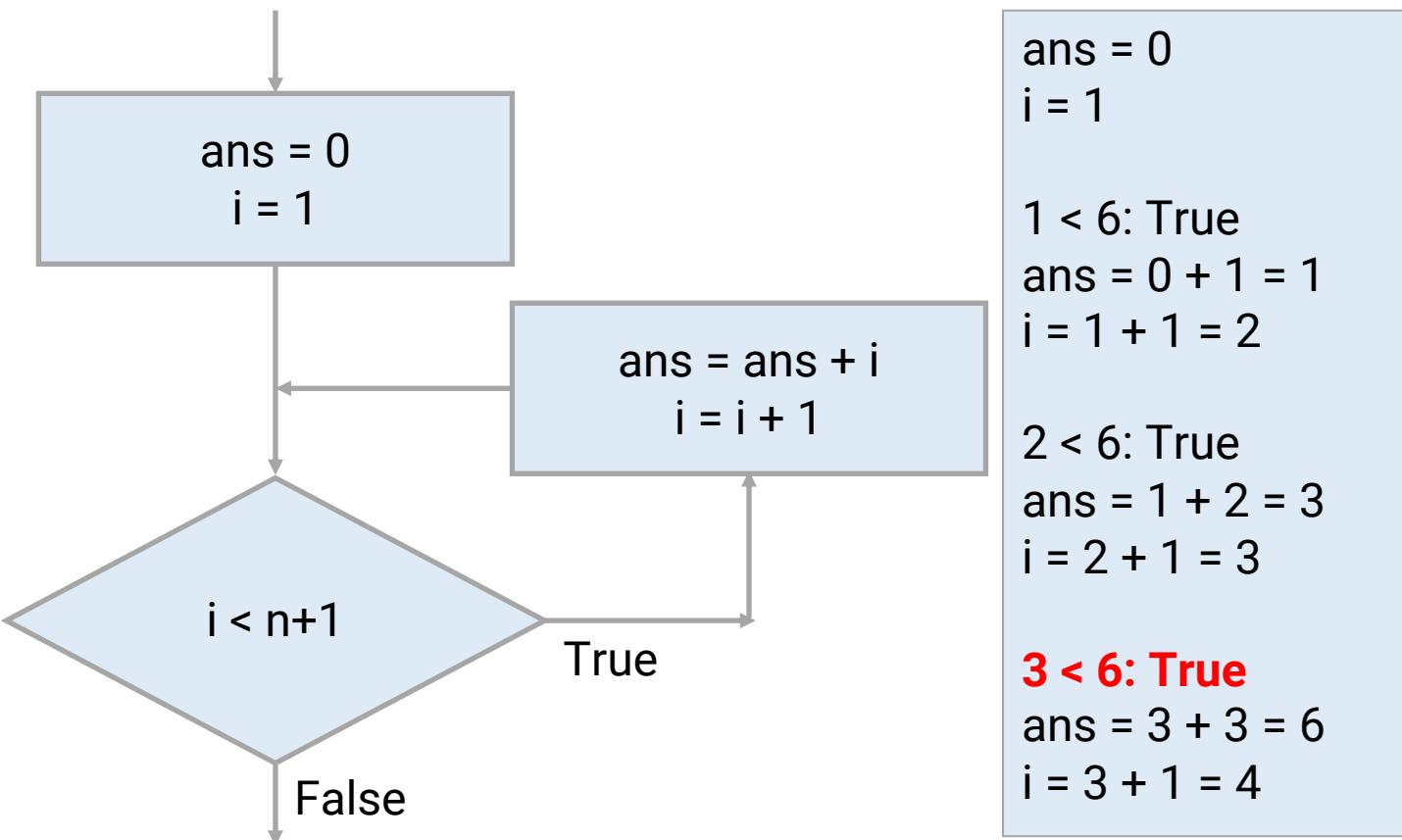
VD: Tính tổng các số từ 1 đến n = 5 – Step by step



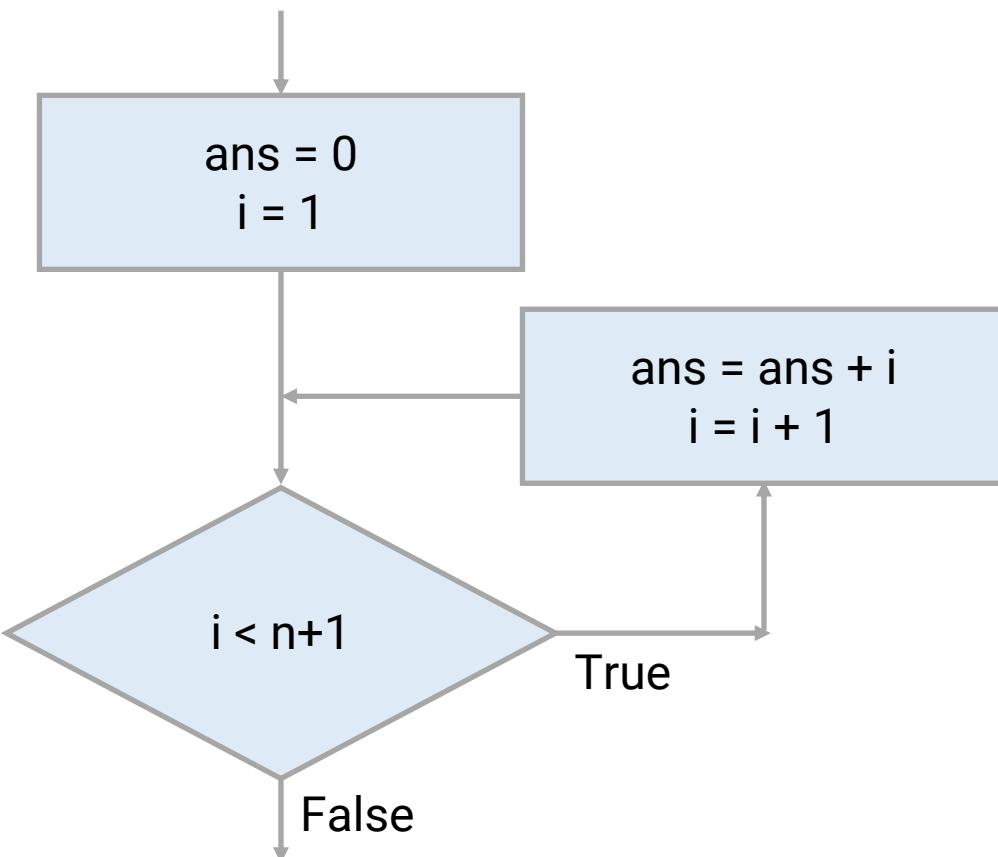
VD: Tính tổng các số từ 1 đến n = 5 – Step by step



VD: Tính tổng các số từ 1 đến n = 5 – Step by step



VD: Tính tổng các số từ 1 đến n = 5 – Step by step



$ans = 0$

$i = 1$

$4 < 6$: True

$ans = 6 + 4 = 10$

$i = 4 + 1 = 5$

$1 < 6$: True

$ans = 0 + 1 = 1$

$i = 1 + 1 = 2$

$2 < 6$: True

$ans = 1 + 2 = 3$

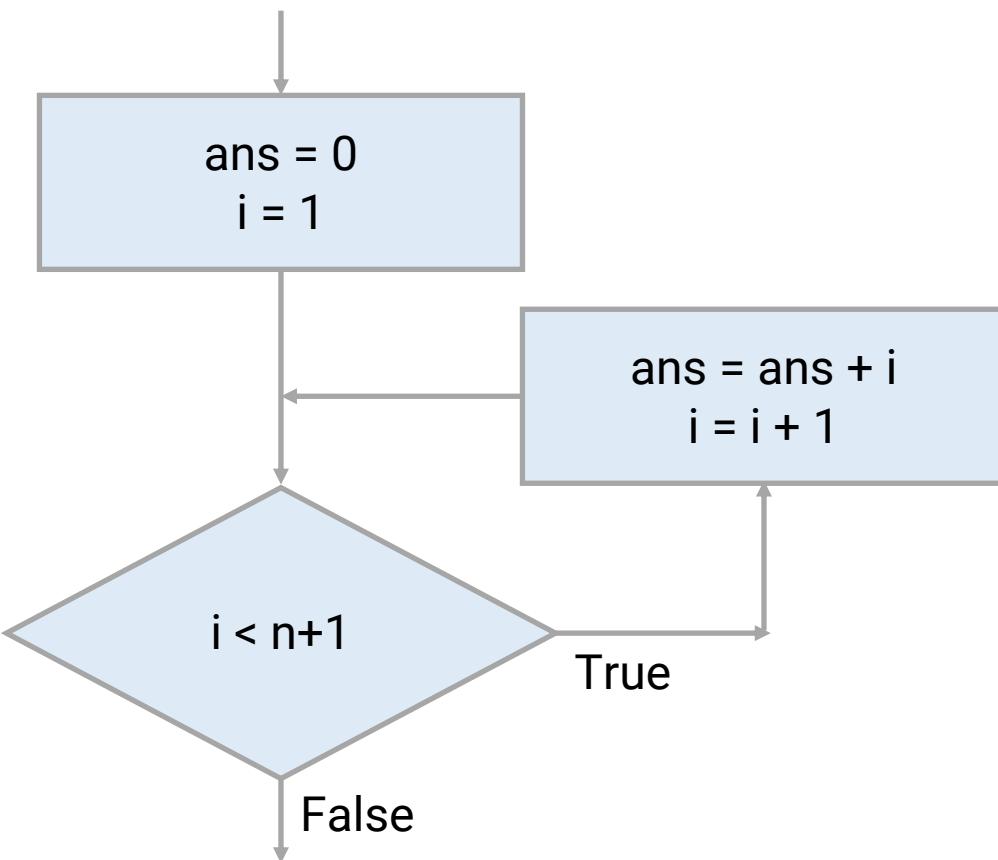
$i = 2 + 1 = 3$

$3 < 6$: True

$ans = 3 + 3 = 6$

$i = 3 + 1 = 4$

VD: Tính tổng các số từ 1 đến n = 5 – Step by step



$ans = 0$

$i = 1$

$1 < 6$: True

$ans = 0 + 1 = 1$

$i = 1 + 1 = 2$

$2 < 6$: True

$ans = 1 + 2 = 3$

$i = 2 + 1 = 3$

$3 < 6$: True

$ans = 3 + 3 = 6$

$i = 3 + 1 = 4$

$4 < 6$: True

$ans = 6 + 4 = 10$

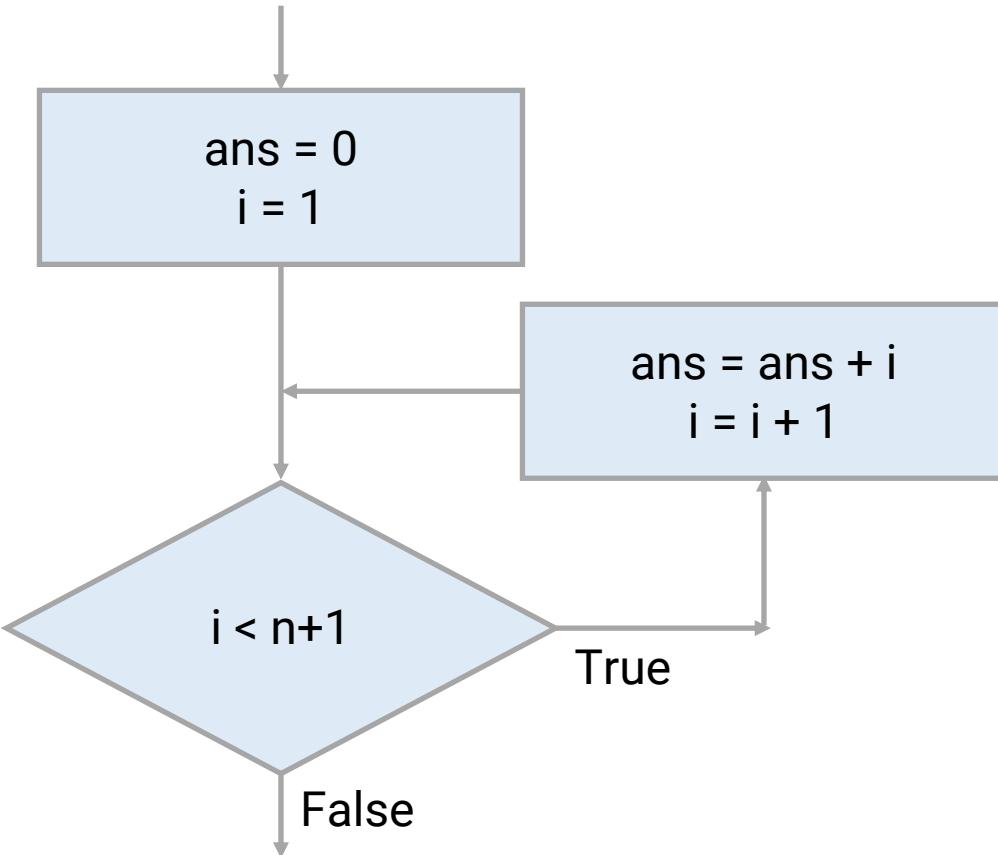
$i = 4 + 1 = 5$

5 < 6: True

$ans = 10 + 5 = 15$

$i = 5 + 1 = 6$

VD: Tính tổng các số từ 1 đến n = 5 – Step by step



$ans = 0$

$i = 1$

$1 < 6$: True

$ans = 0 + 1 = 1$

$i = 1 + 1 = 2$

$2 < 6$: True

$ans = 1 + 2 = 3$

$i = 2 + 1 = 3$

$3 < 6$: True

$ans = 3 + 3 = 6$

$i = 3 + 1 = 4$

$4 < 6$: True

$ans = 6 + 4 = 10$

$i = 4 + 1 = 5$

$5 < 6$: True

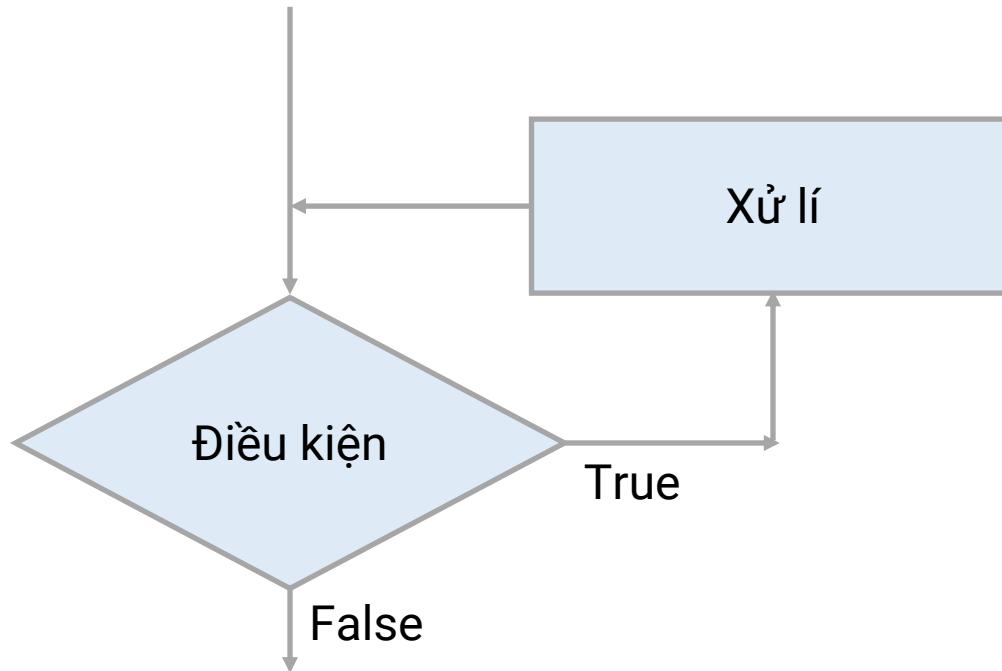
$ans = 10 + 5 = 15$

$i = 5 + 1 = 6$

6 < 6: False

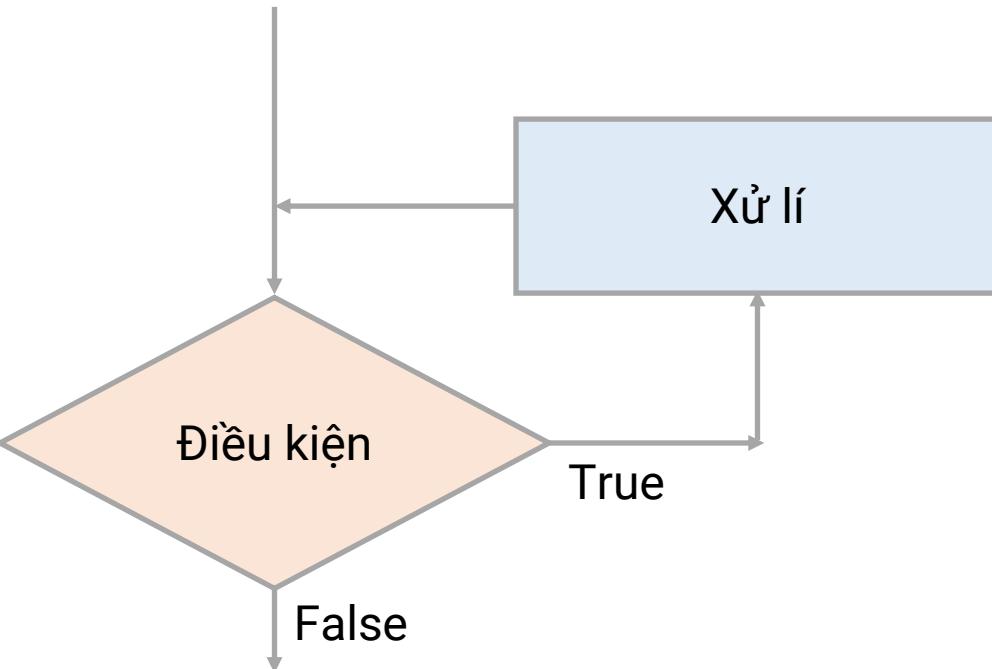
Câu lệnh while

Câu lệnh while



Khi không xác định rõ step

Câu lệnh while

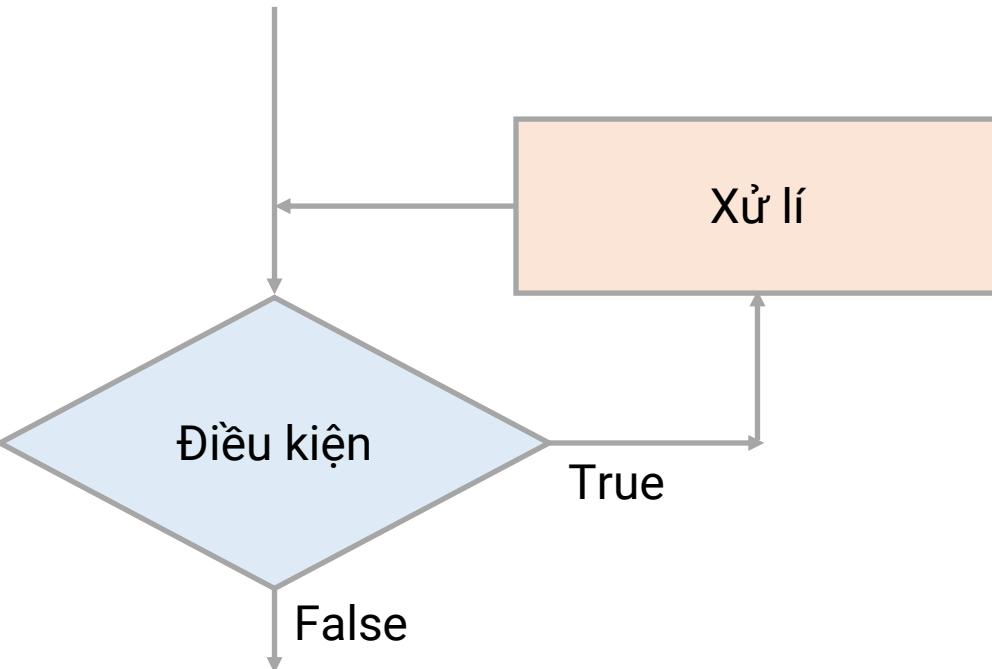


```
while(Điều kiện){  
}
```



```
while(Điều kiện):
```

Câu lệnh while

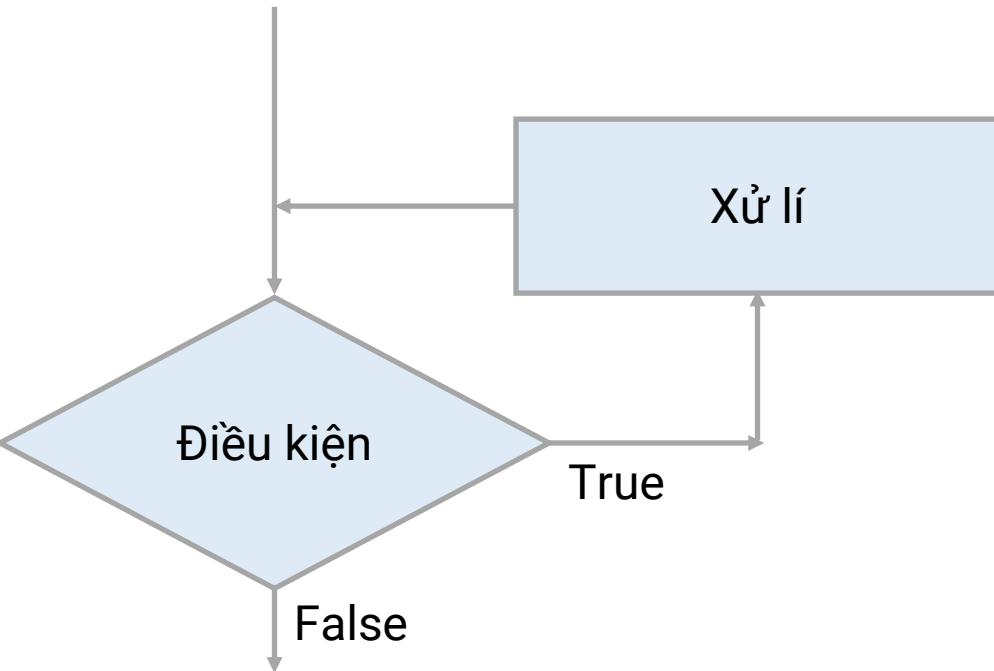


```
while(Điều kiện){  
    Xử lí;  
}
```



```
while(Điều kiện):  
    Xử lí
```

Câu lệnh while



```
while(Điều kiện){
    Xử lí;
}
```

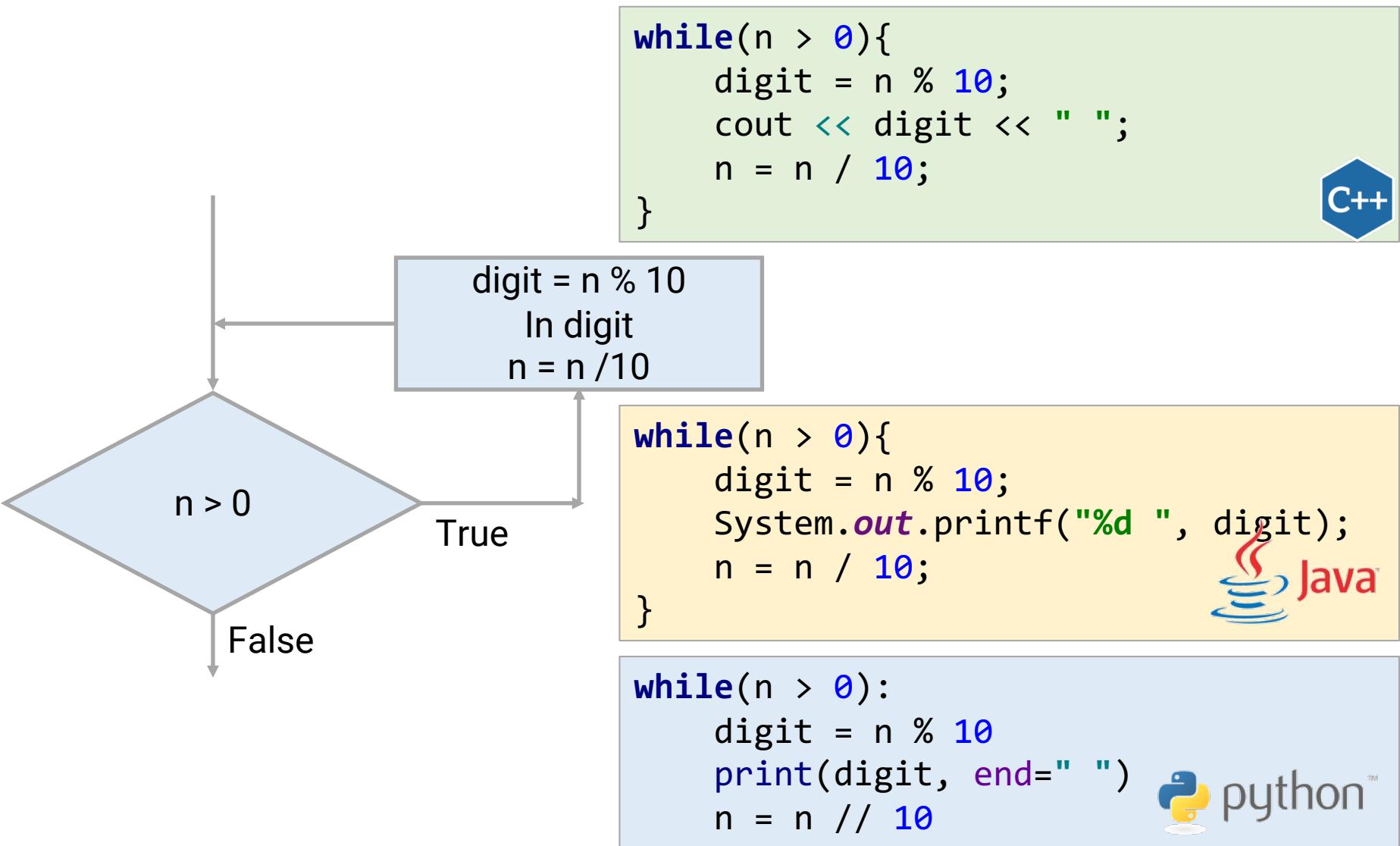
- Cặp dấu {}
- Không có ; ở dòng while
- Không bắt buộc viết lùi “Xử lí” vào 1 tab



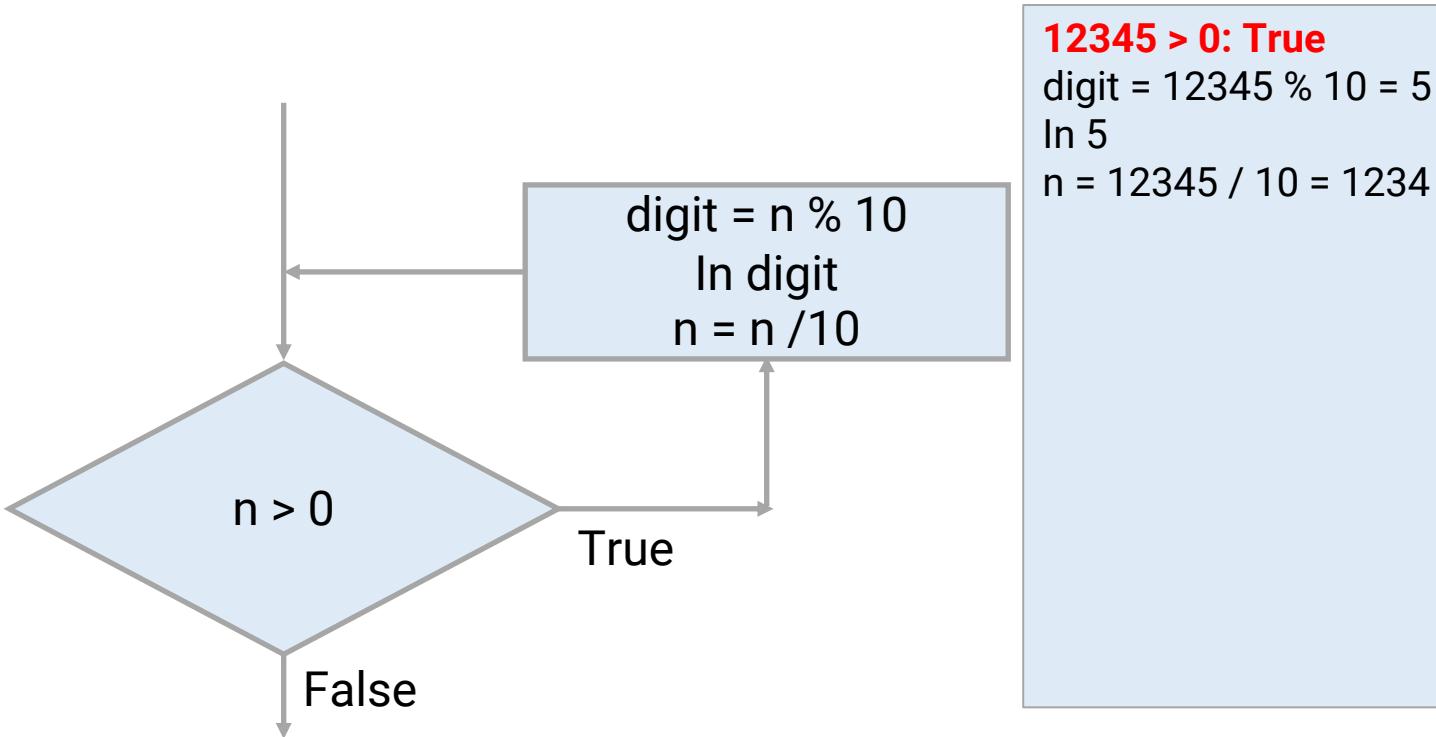
```
while(Điều kiện):
    Xử lí
```

- Không cần cặp dấu {}
- Phải có dấu :
- Bắt buộc viết lùi “Xử lí” vào 1 tab

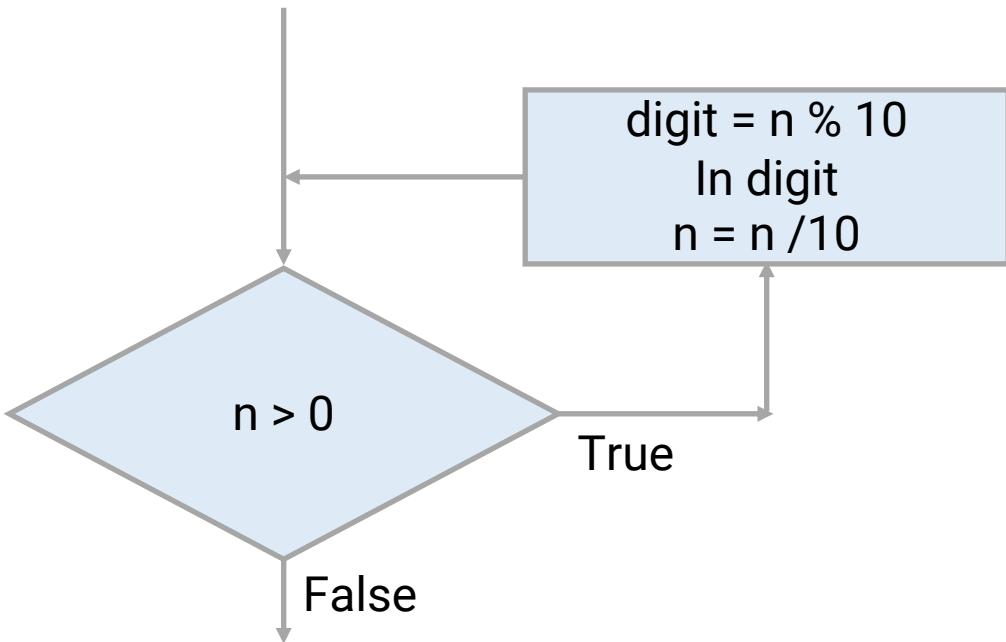
VD: In các chữ số của số nguyên dương n



VD: In các chữ số của số nguyên dương n n = 12345 – Step by step



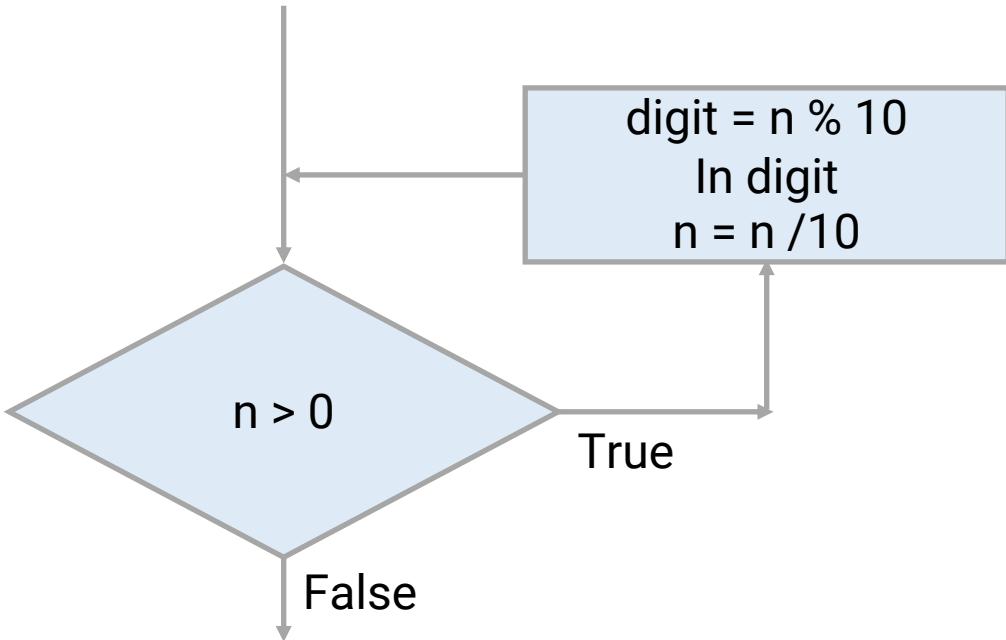
VD: In các chữ số của số nguyên dương n n = 12345 – Step by step



12345 > 0: True
digit = 12345 % 10 = 5
In 5
 $n = 12345 / 10 = 1234$

1234 > 0: True
digit = 1234 % 10 = 4
In 4
 $n = 1234 / 10 = 123$

VD: In các chữ số của số nguyên dương n n = 12345 – Step by step

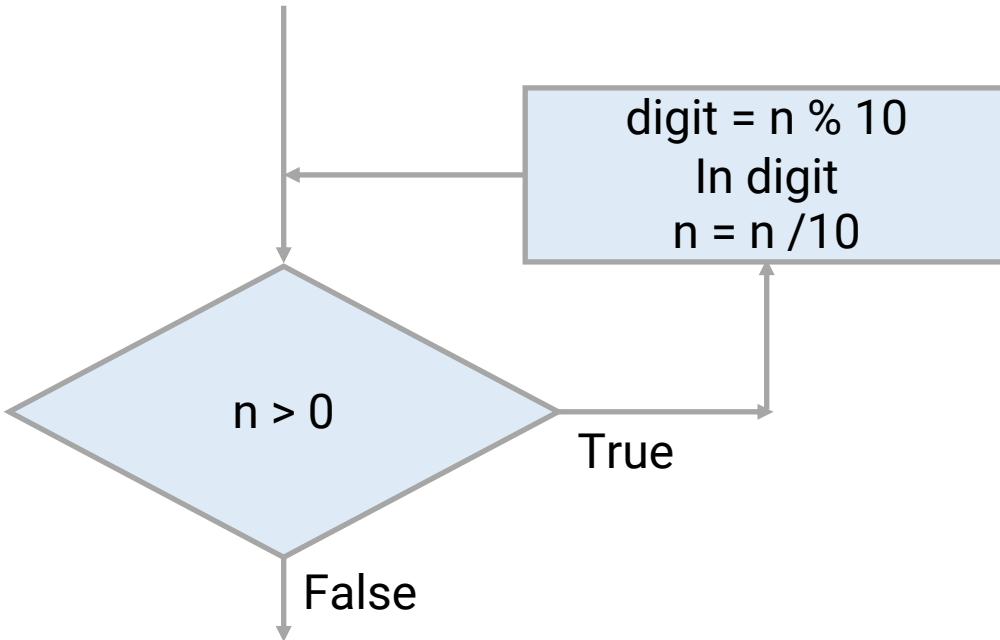


12345 > 0: True
digit = 12345 % 10 = 5
In 5
n = 12345 / 10 = 1234

1234 > 0: True
digit = 1234 % 10 = 4
In 4
n = 1234 / 10 = 123

123 > 0: True
digit = 123 % 10 = 3
In 3
n = 123 / 10 = 12

VD: In các chữ số của số nguyên dương n n = 12345 – Step by step



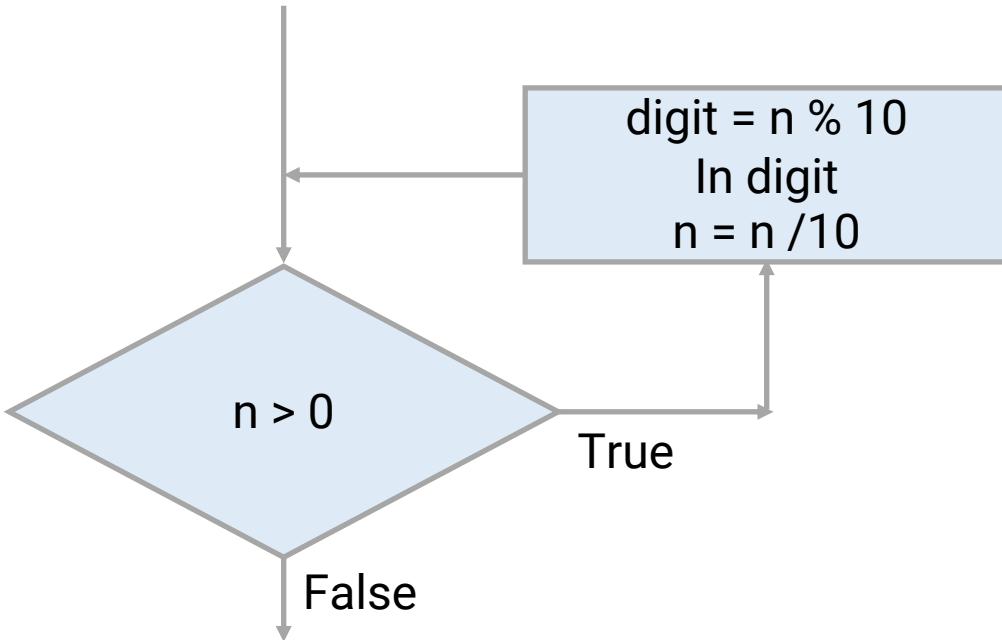
$12345 > 0$: True
 $\text{digit} = 12345 \% 10 = 5$
 $\text{In } 5$
 $n = 12345 / 10 = 1234$

$1234 > 0$: True
 $\text{digit} = 1234 \% 10 = 4$
 $\text{In } 4$
 $n = 1234 / 10 = 123$

$123 > 0$: True
 $\text{digit} = 123 \% 10 = 3$
 $\text{In } 3$
 $n = 123 / 10 = 12$

12 > 0: True
 $\text{digit} = 12 \% 10 = 2$
 $\text{In } 2$
 $n = 12 / 10 = 1$

**VD: In các chữ số của số nguyên dương n
n = 12345 – Step by step**



```
12345 > 0: True  
digit = 12345 % 10 = 5  
In 5  
n = 12345 / 10 = 1234
```

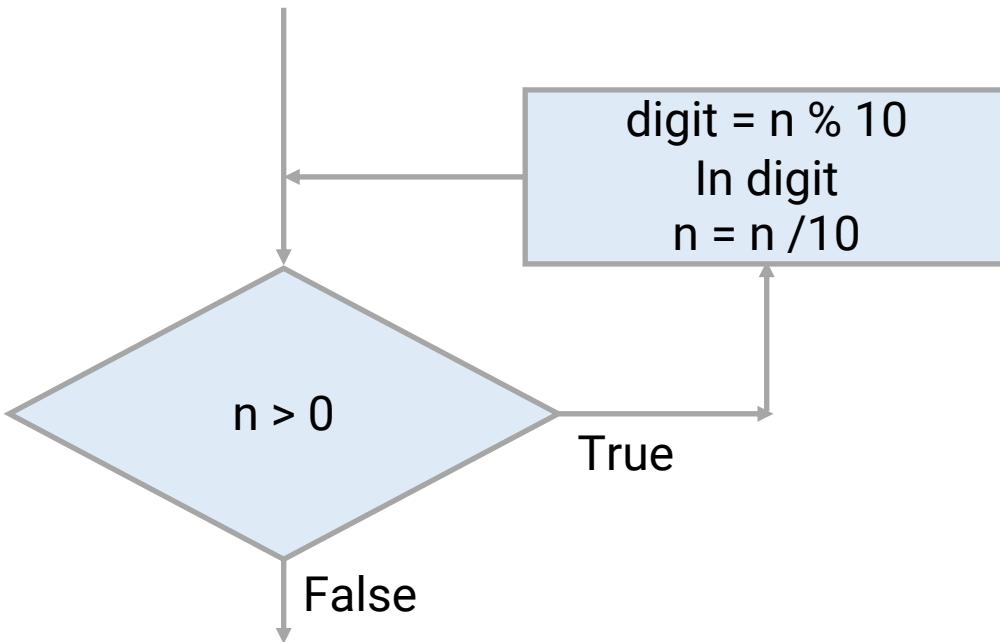
1234 > 0: True
digit = 1234 % 10 = 4
In 4
n = 1234 / 10 = 123

```
123 > 0: True  
digit = 123 % 10 - 3  
In 3  
n = 123 / 10 = 12
```

$12 > 0$: True
 $\text{digit} = 12 \% 10 = 2$
 $\ln 2$
 $n = 12 / 10 = 1$

1 > 0: True
digit = 1 % 10 = 1
ln 1
n = 1 / 10 = 0

VD: In các chữ số của số nguyên dương n n = 12345 – Step by step



$12345 > 0$: True
 $digit = 12345 \% 10 = 5$
 $\ln 5$
 $n = 12345 / 10 = 1234$

$1234 > 0$: True
 $digit = 1234 \% 10 = 4$
 $\ln 4$
 $n = 1234 / 10 = 123$

$123 > 0$: True
 $digit = 123 \% 10 = 3$
 $\ln 3$
 $n = 123 / 10 = 12$

$12 > 0$: True
 $digit = 12 \% 10 = 2$
 $\ln 2$
 $n = 12 / 10 = 1$

$1 > 0$: True
 $digit = 1 \% 10 = 1$
 $\ln 1$
 $n = 1 / 10 = 0$

0 > 0: False

break vs continue

Câu lệnh break

- Câu lệnh **break** làm kết thúc câu lệnh lặp hiện tại.

```
for(int i = 1; i < 10; i++){  
    if(i % 2 == 0){  
        break;  
    }  
    cout << i << " ";  
}
```



```
for i in range(1, 10):  
    if i % 2 == 0:  
        break  
    print(i, end=" ")
```



```
for(int i = 1; i < 10; i++){  
    if(i % 2 == 0){  
        break;  
    }  
    System.out.printf("%d ", i);  
}
```



Lệnh continue

- Lệnh **continue** làm bỏ qua **lần lặp hiện tại**.

```
for(int i = 1; i < 10; i++){  
    if(i % 2 == 0){  
        continue;  
    }  
    cout << i << " ";  
}
```



```
for i in range(1, 10):  
    if i % 2 == 0:  
        continue  
    print(i, end=" ")
```



```
for(int i = 1; i < 10; i++){  
    if(i % 2 == 0){  
        continue;  
    }  
    System.out.printf("%d ", i);  
}
```



1 3 5 7 9

Các kĩ thuật xử lí với vòng lặp

BT1 – TÍNH TỔNG

- Viết chương trình đọc vào số nguyên dương n và tính tổng các số nguyên từ 1 đến n.

Kỹ thuật tính tổng



BT1 – Gợi ý

- Nên sử dụng vòng lặp nào? while hay for?
- Đọc n trước khi vào vòng lặp hay trong vòng lặp?
- Trong vòng lặp, ta phải lặp đi lặp lại phép toán nào?



BT1 – Gợi ý

Lặp đi lặp lại nhiều lần
hành động nào?

1. Sử dụng lệnh nhập, để nhập vào số nguyên n.

3. Sử dụng vòng lặp for i, begin = ?, end = ?, step = ?.
 - a. ans = ans + i.

4. In kết quả ans.

Cộng dồn vào biến ans



BT1 – Gợi ý

Ban đầu,
ans bằng bao nhiêu?

1. Sử dụng lệnh nhập, để nhập vào số nguyên n.
2. Khởi tạo: $\text{ans} = 0$.
3. Sử dụng vòng lặp `for i, begin = ?, end = ?, step = ?.`
 - a. $\text{ans} = \text{ans} + i$.
4. In kết quả ans .

Cộng dồn vào biến ans



BT1 – Gợi ý (Bản đầy đủ)

1. Sử dụng lệnh nhập, để nhập vào số nguyên n.
2. **Khởi tạo: ans = 0.**
3. Sử dụng vòng lặp for i, begin = ?, end = ?, step = ?.
 - a. **ans = ans + i.**
4. In kết quả ans.

Cộng dồn vào biến ans



BT1 – Minh họa

n = 5

ans = 0

i = 1

1 < 6: True

ans = 0 + 1 = 1

i = 1 + 1 = 2

2 < 6: True

ans = 1 + 2 = 3

i = 2 + 1 = 3

3 < 6: True

ans = 3 + 3 = 6

i = 3 + 1 = 4

4 < 6: True

ans = 6 + 4 = 10

i = 4 + 1 = 5

5 < 6: True

ans = 10 + 5 = 15

i = 5 + 1 = 6

6 < 6: False

BT2 – ĐẾM SAO

- Cho danh sách những vật thể trên bầu trời. Bạn hãy giúp Guka đếm xem có bao nhiêu ngôi sao hiện tại có trên bầu trời nhé.
- Ngôi sao là những hình có 5 đỉnh lồi ra, còn lại thì không phải.
- Dữ liệu gồm nhiều dòng, mỗi dòng là một số nguyên dương x ($1 \leq x \leq 100$) là số đỉnh của đối tượng mà Guka nhìn thấy. Cuối bộ dữ liệu vào là một dòng chứa số 0.

Kỹ thuật đếm



BT2 – Gợi ý

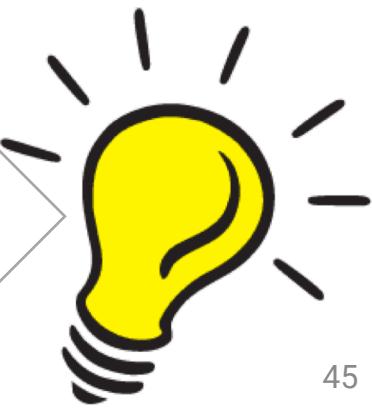
- Nên sử dụng vòng lặp nào? while hay for?
- Đọc x (số cạnh của vật thể) trước khi vào vòng lặp hay trong vòng lặp?
- Trong vòng lặp, ta phải lặp đi lặp lại phép toán nào?
- Điều kiện kết thúc vòng lặp là gì?



BT2 – Gợi ý

2. Sử dụng vòng lặp while(True).

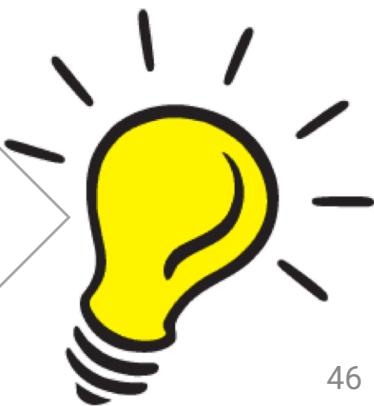
Mỗi lần thỏa điều kiện, hãy tăng count lên 1



BT2 – Gợi ý

2. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc vào số đỉnh x của 1 vật thể.

Mỗi lần thỏa điều kiện, hãy tăng count lên 1



BT2 – Gợi ý

2. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc vào số đỉnh x của 1 vật thể.
 - b. Nếu $x = 0$, kết thúc vòng lặp.

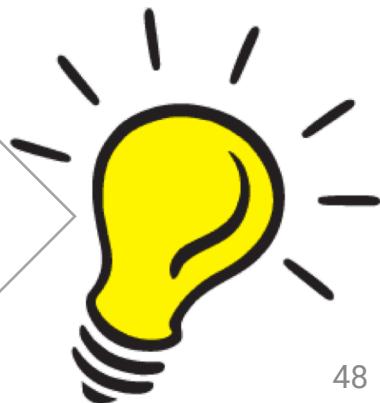
Mỗi lần thỏa điều kiện, hãy tăng count lên 1



BT2 – Gợi ý (Bản đầy đủ)

1. Khởi tạo giá trị đếm count = 0.
2. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc vào số đỉnh x của 1 vật thể.
 - b. Nếu x = 0, kết thúc vòng lặp.
 - c. Nếu x = 5, đây là ngôi sao, tăng biến count lên 1.
3. Sử dụng lệnh xuất, in count.

Mỗi lần thỏa điều kiện, hãy tăng count lên 1



BT2 – Minh họa

```
5  
10  
12  
5  
0
```

```
count = 0  
  
x = 5  
x == 0: False  
x == 5: True, count = 0 + 1 = 1  
  
x = 10  
x == 0: False  
x == 5: False  
  
x = 12  
x == 0: False  
x == 5: False  
  
x = 5  
x == 0: False  
x == 5: True, count = 1 + 1 = 2  
  
x = 0  
x == 0: True
```

BT3 – SO SÁNH

- Hôm nay cô giáo phát bài kiểm tra trong lớp. Cô đọc điểm từng học sinh lần lượt.
- Sau khi đọc cô muốn bạn thống kê điểm số cao nhất lớp và điểm số thấp nhất lớp.

Kĩ thuật đặt lính canh



BT3 – Gợi ý

- Nên sử dụng vòng lặp nào? while hay for?
- Đọc x (điểm số của một học sinh) trước khi vào vòng lặp hay trong vòng lặp?
- Giả sử min và max là 2 biến lưu điểm thấp nhất và cao nhất. Trong vòng lặp, khi nào update min, khi nào update max?
- Điều kiện kết thúc vòng lặp là gì?



BT3 – Gợi ý

Trong vòng lặp, cập nhật max/min khi gặp giá trị lớn hơn/nhỏ hơn lính canh

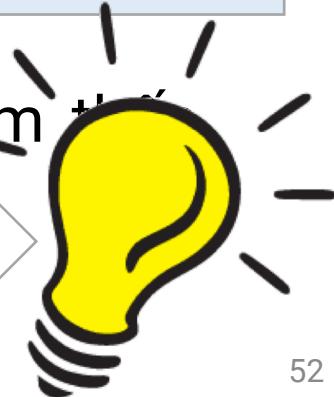
2. Sử dụng vòng lặp while True.

- c. Nếu $x >$ điểm cao nhất hiện tại max, cập nhật $\text{max} = x$.
- d. Nếu $x <$ điểm thấp nhất hiện tại min, cập nhật $\text{min} = x$.

3. Sử dụng lệnh xuất, in điểm cao nhất và điểm thấp nhất.

Đặt giá trị ban đầu cho lính canh.

Nếu giá trị hiện tại “tốt hơn”, thì cập nhật lính canh.



BT3 – Gợi ý

x ở đâu ra?

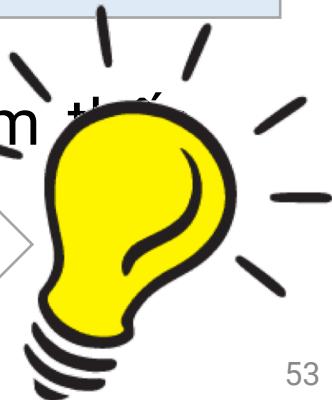
2. Sử dụng vòng lặp while True.

- a. Sử dụng lệnh nhập, đọc vào x là điểm 1 học sinh.
- c. Nếu $x >$ điểm cao nhất hiện tại max, cập nhật $\text{max} = x$.
- d. Nếu $x <$ điểm thấp nhất hiện tại min, cập nhật $\text{min} = x$.

3. Sử dụng lệnh xuất, in điểm cao nhất và điểm thấp nhất.

Đặt giá trị ban đầu cho lính canh.

Nếu giá trị hiện tại “tốt hơn”, thì cập nhật lính canh.



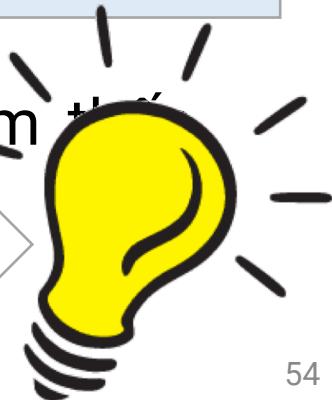
BT3 – Gợi ý

max/min ban đầu
bằng bao nhiêu?

1. Khởi tạo 2 lính canh max = 10, min = 0.
2. Sử dụng vòng lặp while True.
 - a. Sử dụng lệnh nhập, đọc vào x là điểm 1 học sinh.
 - b. ...
 - c. Nếu $x >$ điểm cao nhất hiện tại max, cập nhật max = x.
 - d. Nếu $x <$ điểm thấp nhất hiện tại min, cập nhật min = x.
3. Sử dụng lệnh xuất, in điểm cao nhất và điểm thấp nhất.

Đặt giá trị ban đầu cho lính canh.

Nếu giá trị hiện tại “tốt hơn”, thì cập nhật lính canh.



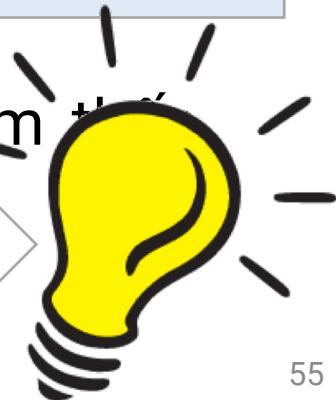
BT3 – Gợi ý

Nếu bạn làm tốt nhất của lớp
chỉ có 9.5đ, thì code này in ra
max bao nhiêu?

1. Khởi tạo 2 lính canh max = -1, min = 11.
2. Sử dụng vòng lặp while True.
 - a. Sử dụng lệnh nhập, đọc vào x là điểm 1 học sinh.
 - b. ...
 - c. Nếu $x >$ điểm cao nhất hiện tại max, cập nhật max = x.
 - d. Nếu $x <$ điểm thấp nhất hiện tại min, cập nhật min = x.
3. Sử dụng lệnh xuất, in điểm cao nhất và điểm thấp nhất.

Đặt giá trị ban đầu cho lính canh.

Nếu giá trị hiện tại “tốt hơn”, thì cập nhật lính canh.



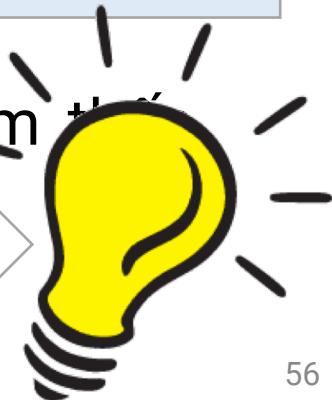
BT3 – Gợi ý

Vòng lặp kết thúc khi nào?

1. Khởi tạo 2 lính canh max = -1, min = 11.
2. Sử dụng vòng lặp while True.
 - a. Sử dụng lệnh nhập, đọc vào x là điểm 1 học sinh.
 - b. Nếu x = -1, kết thúc vòng lặp.
 - c. Nếu x > điểm cao nhất hiện tại max, cập nhật max = x.
 - d. Nếu x < điểm thấp nhất hiện tại min, cập nhật min = x.
3. Sử dụng lệnh xuất, in điểm cao nhất và điểm thấp nhất.

Đặt giá trị ban đầu cho lính canh.

Nếu giá trị hiện tại “tốt hơn”, thì cập nhật lính canh.

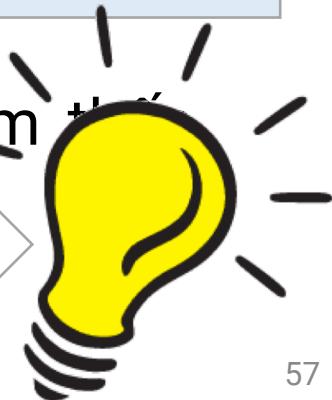


BT3 – Gợi ý (Bản đầy đủ)

1. Khởi tạo 2 lính canh max = -1, min = 11.
2. Sử dụng vòng lặp while True.
 - a. Sử dụng lệnh nhập, đọc vào x là điểm 1 học sinh.
 - b. Nếu x = -1, kết thúc vòng lặp.
 - c. Nếu x > điểm cao nhất hiện tại max, cập nhật max = x.
 - d. Nếu x < điểm thấp nhất hiện tại min, cập nhật min = x.
3. Sử dụng lệnh xuất, in điểm cao nhất và điểm thấp nhất.

Đặt giá trị ban đầu cho lính canh.

Nếu giá trị hiện tại “tốt hơn”, thì cập nhật lính canh.



BT3 – Minh họa

```
2  
7  
9  
1  
-1
```

max = -1
min = 11

x = 2
x == -1: False
x = 2 > max = -1: True, max = 2
x = 2 < min = 11: True, min = 2

x = 7
x == -1: False
x = 7 > max = 2: True, max = 7
x = 7 < min = 2: False

x = 9
x == -1: False
x = 9 > max = 7: True, max = 9
x = 9 < min = 2: False

x = 1
x == -1: False
x = 1 > max = 9: False
x = 1 < lowest = 2: True, min = 1

x = -1
x == -1: **True**

BT4 – SỐ NGUYÊN TỐ

- Kiểm tra số nguyên dương n có phải là số nguyên tố hay không.
- Số nguyên tố là số chỉ chia hết cho 1 và chính nó.

Kĩ thuật đếm

Kĩ thuật đặt lính canh



BT4 – Gợi ý

- Có 2 cách giải bài này:

- C1: Số nguyên tố là số có 2 ước số là 1 và chính nó → dùng kĩ thuật đếm, cho i chạy từ 1 đến n, đếm số lượng ước số i ($n \% i == 0$) của n. Nếu count == 2 thì n là số nguyên tố.
- C2: Số nguyên tố là số chỉ chia hết cho 1 và chính nó → dùng kĩ thuật đặt cờ hiệu, cho i chạy từ 2 đến n-1, tìm 1 trường hợp ngoại lệ $n \% i == 0$. Nếu có ngoại lệ, thì i ko phải là số nguyên tố. Ngược lại, nếu ko có ngoại lệ, thì i là số nguyên tố.



BT4 – Gợi ý (C2) (Bản đầy đủ)

1. Sử dụng lệnh nhập, đọc số nguyên dương n.
2. Khởi tạo **flag = True** (tin tưởng n là số nguyên tố)
3. Sử dụng vòng lặp, i từ 2 đến n-1
 - a. Nếu $n \% i == 0$, đánh dấu **flag = False** (n ko phải là số nguyên tố)
4. Kết thúc vòng lặp, kiểm tra nếu flag == True thì n là số nguyên tố.
5. Nếu không, n không phải là số nguyên tố.



BT4 – Minh họa ($n = 6$)

6

$n = 6$

$i = 2$

flag = True

$2 < 6$: True

$6 \% 2 == 0$: True

flag = False

$i = 2 + 1 = 3$

$3 < 6$: True

$6 \% 3 == 0$: True

flag = False

$i = 3 + 1 = 4$

$4 < 6$: True

$6 \% 4 == 0$: False

$i = 4 + 1 = 5$

$5 < 6$: True

$6 \% 5 == 0$: False

$i = 5 + 1 = 6$

$6 < 6$: False

BT5 – DÃY TĂNG

- Lulu có một vườn hoa có n chậu. Mỗi chậu hoa có một độ cao nhất định.
- Tết đến rồi Lulu mong muốn có một vườn hoa mặt trời như ý. Một vườn hoa mà Lulu cảm thấy thích là khi các chậu hoa phía sau chắc chắn phải cao hơn hoặc bằng chậu hoa ở phía trước.
- Lulu thuê những người thợ làm vườn tới để tỉa chậu hoa theo ý của anh ấy. Nhưng hôm nay anh ấy có việc phải ra ngoài, anh ấy nhờ bạn **kiểm tra, liệu với độ cao những chậu hoa sau khi tỉa thì có phù hợp với yêu cầu của anh ấy hay không?**

Kĩ thuật đặt cờ hiệu



BT5 – Gợi ý

- Nên sử dụng vòng lặp nào? while hay for?
- Đọc hcur (chiều cao của cây hiện tại) trước khi vào vòng lặp hay trong vòng lặp?
- Giả sử hprev và hcur là chiều cao cây trước đó và chiều cao cây hiện tại. Khi nào thì hprev và hcur vi phạm qui tắc dãy tăng?
- Điều kiện kết thúc vòng lặp là gì?



BT5 – Gợi ý

Đọc dữ liệu nhập vào

2. Khởi tạo cờ hiệu flag = True.
3. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc chiều cao cây hiện tại, hcur.
4. In kết quả dựa vào giá trị True/False của biến flag.

**Tìm 1 ví dụ ngược lại, để chứng minh
phát biểu đưa ra là không đúng**

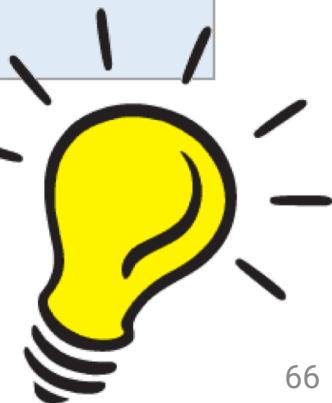


BT5 – Gợi ý

Trong vòng lặp,
tìm 1 trường hợp
cây sau “lùn hơn” cây trước

2. Khởi tạo cờ hiệu flag = True.
3. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc chiều cao cây hiện tại, hcur.
 - c. Nếu chiều cao cây hiện tại hcur < chiều cao cây trước đó hprev, đặt lại cờ hiệu flag = False.
4. In kết quả dựa vào giá trị True/False của biến flag.

Tìm 1 ví dụ ngược lại, để chứng minh
phát biểu đưa ra là không đúng

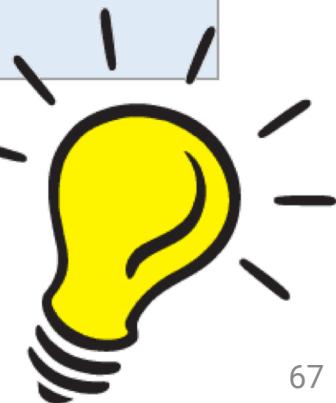


BT5 – Gợi ý

Làm sao đảm bảo
hprev là cây trước
của cây hcur?

2. Khởi tạo cờ hiệu flag = True.
3. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc chiều cao cây hiện tại, hcur.
 - c. Nếu chiều cao cây hiện tại hcur < chiều cao cây trước đó hprev, đặt lại cờ hiệu flag = False.
 - d. hprev = hcur.
4. In kết quả dựa vào giá trị True/False của biến flag.

Tìm 1 ví dụ ngược lại, để chứng minh
phát biểu đưa ra là không đúng



BT5 – Gợi ý

Lúc đọc vào chiều
cao cây đầu tiên, thì
hprev là bao nhiêu?

1. Khởi tạo hprev = 999.
2. Khởi tạo cờ hiệu flag = True.
3. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc chiều cao cây hiện tại, hcur.
 - c. Nếu chiều cao cây hiện tại hcur < chiều cao cây trước đó hprev, đặt lại cờ hiệu flag = False.
 - d. hprev = hcur.
4. In kết quả dựa vào giá trị True/False của biến flag.

Tìm 1 ví dụ ngược lại, để chứng minh
phát biểu đưa ra là không đúng

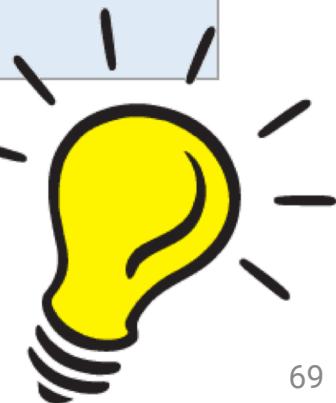


BT5 – Gợi ý

Xét các cây 1 2 3 4 5.
Cây đầu tiên là 1, thì $1 < 999$, vậy flag = False
đúng ko?

1. Khởi tạo hprev = -1. (ko có cây nào có chiều cao âm)
2. Khởi tạo cờ hiệu flag = True.
3. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc chiều cao cây hiện tại, hcur.
 - b. Nếu chiều cao cây hiện tại hcur < chiều cao cây trước đó hprev, đặt lại cờ hiệu flag = False.
 - c. hprev = hcur.
4. In kết quả dựa vào giá trị True/False của biến flag.

Tìm 1 ví dụ ngược lại, để chứng minh
phát biểu đưa ra là không đúng

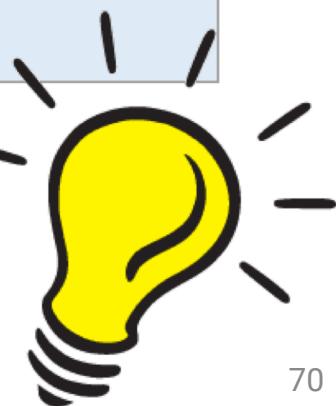


BT5 – Gợi ý

Khi nào vòng lặp kết thúc?

1. Khởi tạo $h_{prev} = -1$. (ko có cây nào có chiều cao âm)
2. Khởi tạo cờ hiệu $flag = True$.
3. Sử dụng vòng lặp $while(True)$.
 - a. Sử dụng lệnh nhập, đọc chiều cao cây hiện tại, h_{cur} .
 - b. Nếu chiều cao cây hiện tại $h_{cur} = 0$, kết thúc vòng lặp.
 - c. Nếu chiều cao cây hiện tại $h_{cur} <$ chiều cao cây trước đó h_{prev} , đặt lại cờ hiệu $flag = False$.
 - d. $h_{prev} = h_{cur}$.
4. In kết quả dựa vào giá trị True/False của biến $flag$.

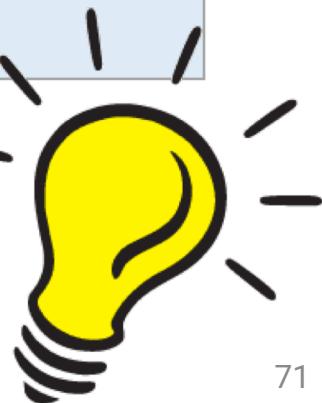
Tìm 1 ví dụ ngược lại, để chứng minh
phát biểu đưa ra là không đúng



BT5 – Gợi ý (Bản đầy đủ)

1. Khởi tạo hprev = -1. (ko có cây nào có chiều cao âm)
2. Khởi tạo cờ hiệu **flag = True**.
3. Sử dụng vòng lặp while(True).
 - a. Sử dụng lệnh nhập, đọc chiều cao cây hiện tại, hcur.
 - b. Nếu chiều cao cây hiện tại hcur = 0, kết thúc vòng lặp.
 - c. Nếu chiều cao cây hiện tại hcur < chiều cao cây trước đó hprev, **đặt lại cờ hiệu flag = False**.
 - d. hprev = hcur.
4. In kết quả dựa vào giá trị True/False của biến flag.

**Tìm 1 ví dụ ngược lại, để chứng minh
phát biểu đưa ra là không đúng**



BT5 – Minh họa

```
1  
2  
3  
1  
0
```

hprev = -1
flag = True

hcur = 1
hcur == 0: False
hcur = 1 < hprev = -1: False
hprev = hcur = 1

hcur = 2
hcur == 0: False
hcur = 2 < hprev = 1: False
hprev = hcur = 2

hcur = 3
hcur == 0: False
hcur = 3 < hprev = 2: False
hprev = hcur = 3

hcur = 1
hcur == 0: False
hcur = 1 < hprev = 3: True, flag = False
hprev = hcur = 1

hcur = 0
hcur == 0: **True**

Lưu ý trên C++, Java

Thiếu 1 trong 3 thành phần trong for

- Có thể thiếu 1 trong 3 thành phần: begin, end, step, nhưng phải giữ lại 2 dấu ;

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

```
for(int i = 5;; i++){
    if(i >= 11)
        break;
    cout << i << " ";
}
```

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

```
int i = 5;
for(;i < 11; i++){
    System.out.printf("%d ", i);
}
```

```
for(int i = 5;; i++){
    if(i >= 11)
        break;
    System.out.printf("%d ", i);
}
```

```
for(int i = 5;i < 11;) {
    System.out.printf("%d ", i);
    i++;
}
```

Nhiều begin, nhiều step trong for

- C++: Có thể có nhiều khởi tạo, nhiều bước nhảy, cách nhau dấu phẩy.
- Java: lỗi biên dịch.

```
for(int i = 1, j = 0; i < 11; i++, j+= 2){  
    cout << i * j << " ";  
}
```

```
0 4 12 24 40 60 84 112 144 180
```

Cẩn thận vòng lặp vô tận!!!

```
for(int i = 0; i < 11; i--){
    cout << i << " ";
}
```

```
int i = 0;
while(i < 11){
    cout << i << " ";
}
```

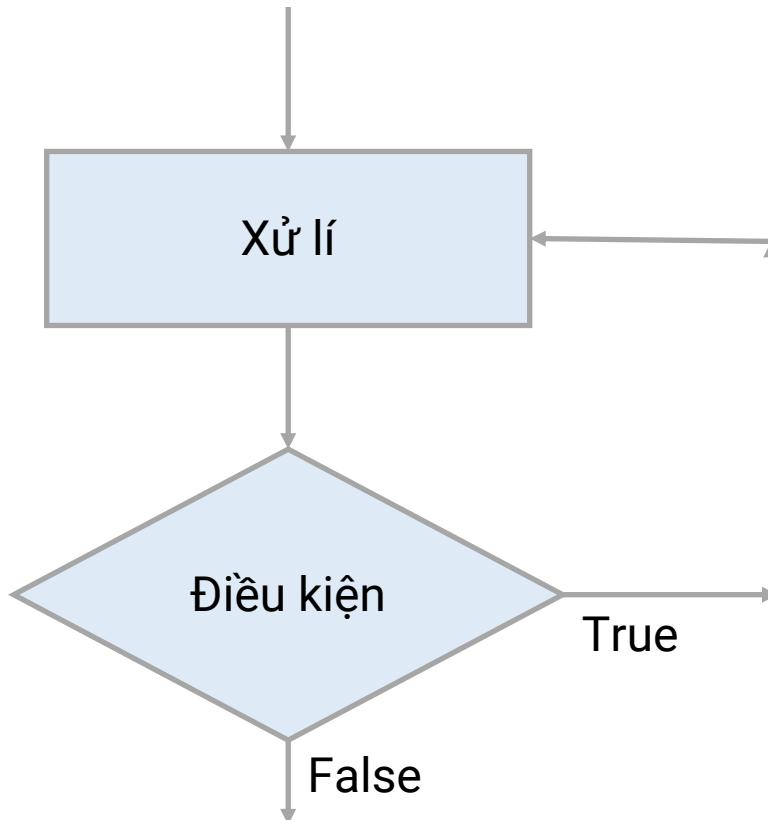
```
int i = 0;
while(i < 11);
{
    cout << i << " ";
    i++;
}
```

```
for(int i = 0; i < 11; i--){
    System.out.printf("%d ", i);
}
```

```
int i = 0;
while(i < 11){
    System.out.printf("%d ", i);
}
```

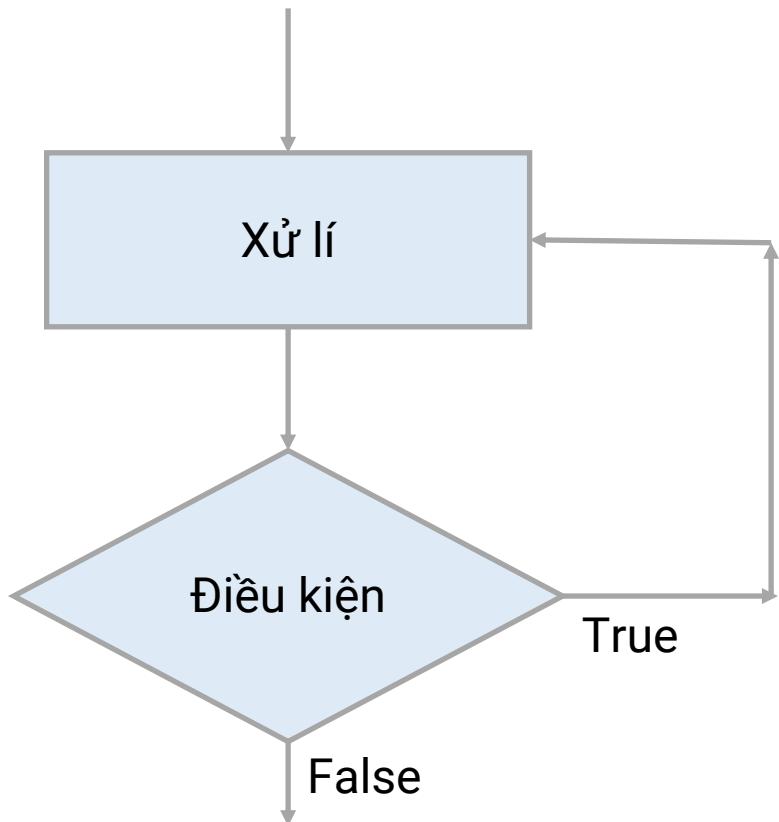
```
int i = 0;
while(i < 11);
{
    System.out.printf("%d ", i);
    i++;
}
```

Câu lệnh do ... while trong C++, Java



Khi cần xử lí trước, kiểm tra điều kiện sau

Câu lệnh do ... while trong C++, Java

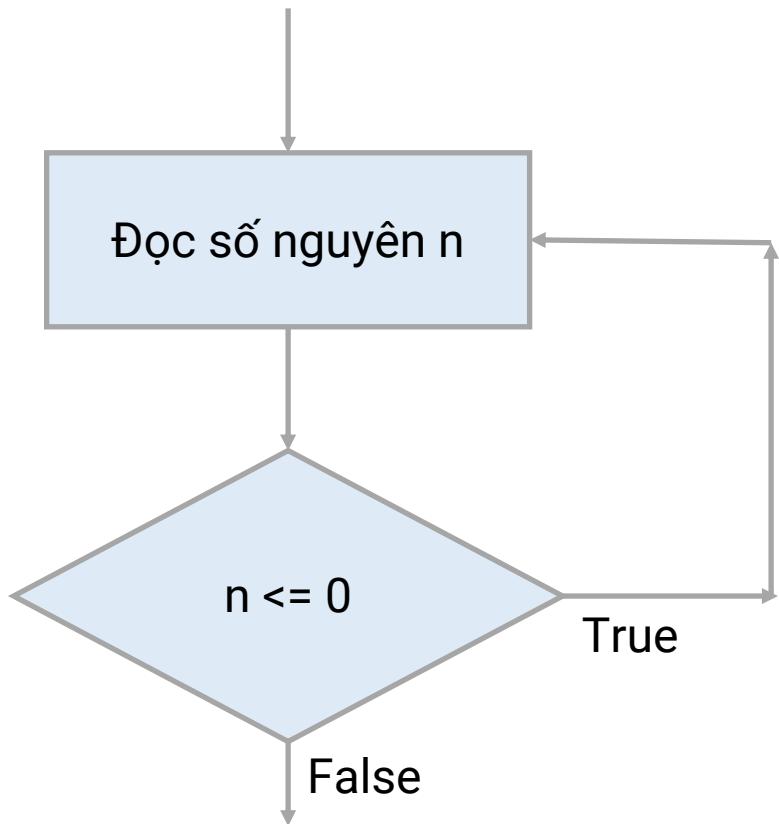


```
do{  
    Xử lí;  
}while(Điều kiện);
```

- Cặp dấu {}
- Không bắt buộc viết lùi “Xử lí” vào 1 tab
- Có ; ở dòng while
- Thường viết } và while(Điều kiện); trên cùng 1 dòng.

Python ko hỗ trợ

Ví dụ



```
// Đọc số nguyên dương n
int n;
do{
    cin >> n;
}while(n <= 0);
```

```
// Đọc số nguyên dương n
int n;
do{
    n = sc.nextInt();
}while(n <= 0);
```

```
n = 0
while(n <= 0):
    n = int(input())
```

Lưu ý trên Python

range(end)

```
for i in range(5):  
    print(i)
```

```
0  
1  
2  
3  
4
```

- Lưu ý:
 - begin = 0.
 - end = 5 → $i < 5$ → chỉ in đến 4.
 - step = 1.

range(begin, end)

```
for i in range(1, 5):  
    print(i)
```

```
1  
2  
3  
4
```

- Lưu ý:
 - step = 1.

range(begin, end, step)

```
for i in range(1, 10, 2):  
    print(i)
```

```
1  
3  
5  
7  
9
```

- Lưu ý:
 - step = 2.
 - In ra các số lẻ từ 1 và nhỏ hơn 10.

range(begin, end, step) – empty

```
for i in range(10, 1):  
    print(i)
```

```
# Empty
```

- Lưu ý:
 - $i = \text{begin} = 10.$
 - $\text{end} = 1.$
 - $i < \text{end} \rightarrow \text{False!!!}$

range(begin, end, step) – step < 0

```
for i in range(5, 1, -1):  
    print(i)
```

```
5  
4  
3  
2
```

- Lưu ý:
 - begin = 5.
 - end = 1 → i > 1.
 - step = -1 → i -= 1 → i giảm dần.

range() – float numbers

```
for i in range(0.5, 1.0, 0.1):  
    print(i)
```

Error

- range() không hỗ trợ kiểu float.

Vòng lặp vô tận

```
# DO NOT RUN THIS CODE!!!!  
while True:  
    print ('Uh Oh infinite Loop!')
```

```
Uh Oh infinite Loop!  
...  
...
```

Hỏi đáp

