

NHẬP MÔN LẬP TRÌNH (23CLC05)

Buổi 3 – 16/10/23

Chương trình khác tiến trình thế nào? (1)

Lưu đồ (Flowchart)	Python
<pre>graph TD; Start([Start]) --> Input[/Nhập a, b/]; Input --> Process[r = a mod b]; Process --> Output[/Xuất r/]; Output --> End([End]);</pre>	<pre>a = int(input("Enter an integer a: ")) b = int(input("Enter an integer b: ")) print("a mod b = ", a % b) print("Done!")</pre>
	C/C++
	<pre>#include <stdio.h> int main() { int a, b; printf("Enter 2 integers a b: "); scanf("%d %d", &a, &b); printf("a mod b = %d\n", a % b); printf("Done!\n"); }</pre>

Chương trình khác tiến trình thế nào? (2)

- **Tiến trình** (process) là một quá trình thực thi cụ thể của **chương trình** (program)
- Chương trình là kịch bản thực thi của tất cả các tiến trình; còn tiến trình là trường hợp thực thi cụ thể của một trong các khả năng mà chương trình qui định

Luồng thực thi là gì?

- **Luồng thực thi** (execution flow, control flow) là thứ tự thực thi các lệnh của một chương trình
- Dạng luồng thực thi căn bản và đơn giản nhất là thực thi **tuần tự** (sequential) mỗi lệnh đúng một lần từ đầu đến cuối chương trình
 - trừ khi một lệnh nào đó có lỗi thực thi thì tiến trình sẽ dừng mà không thực thi các lệnh sau đó

Lưu đồ là gì?

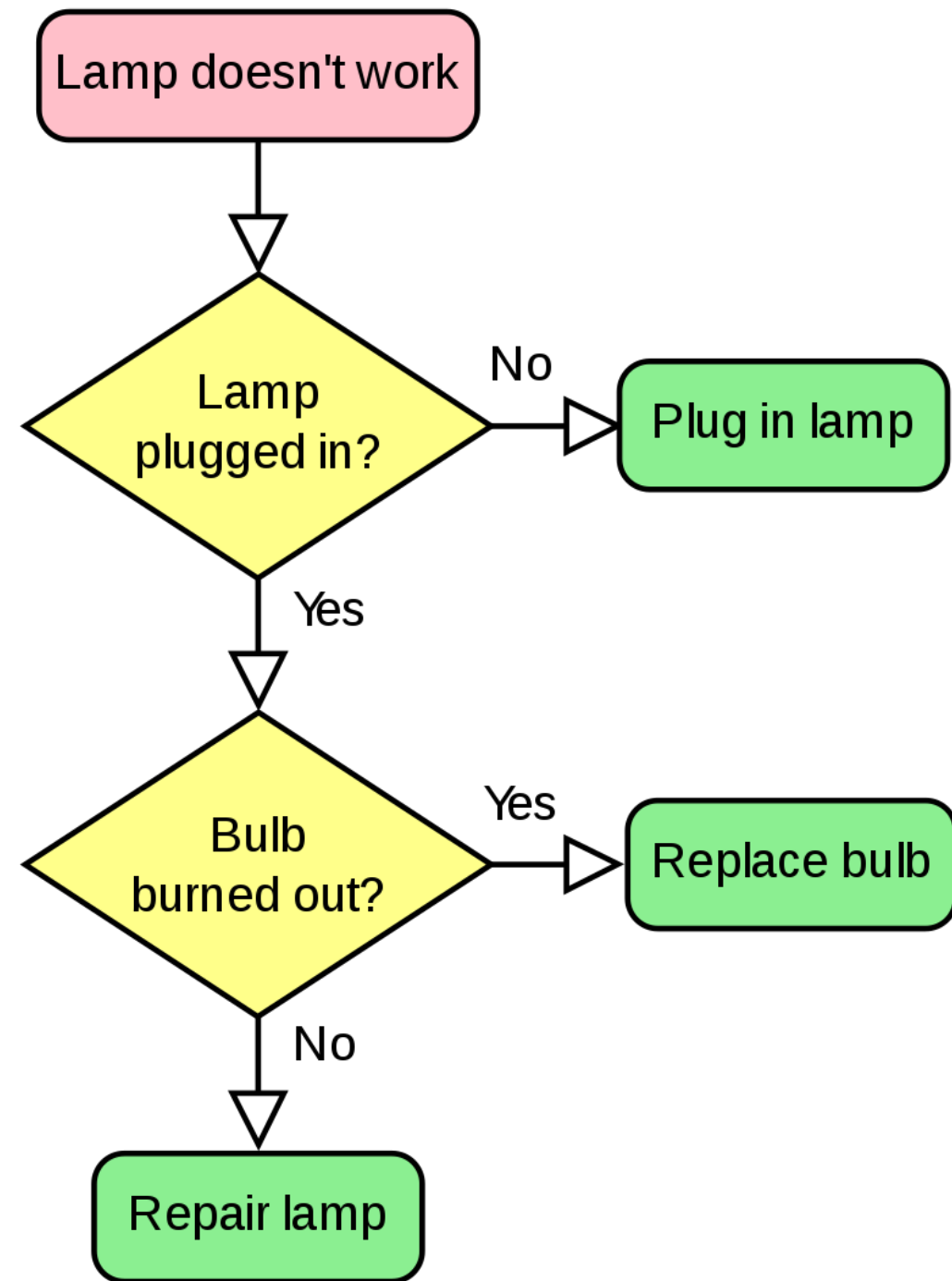
- **Lưu đồ** (flowchart) là sơ đồ mô tả luồng thực thi của một công việc hay chương trình
- Các bước hay các công đoạn được mô tả bằng các **nút** (node, box) và thứ tự thực hiện được mô tả bằng các **cạnh có hướng** (directed edge, arrow)

Các loại nút nào hay dùng trong lưu đồ?

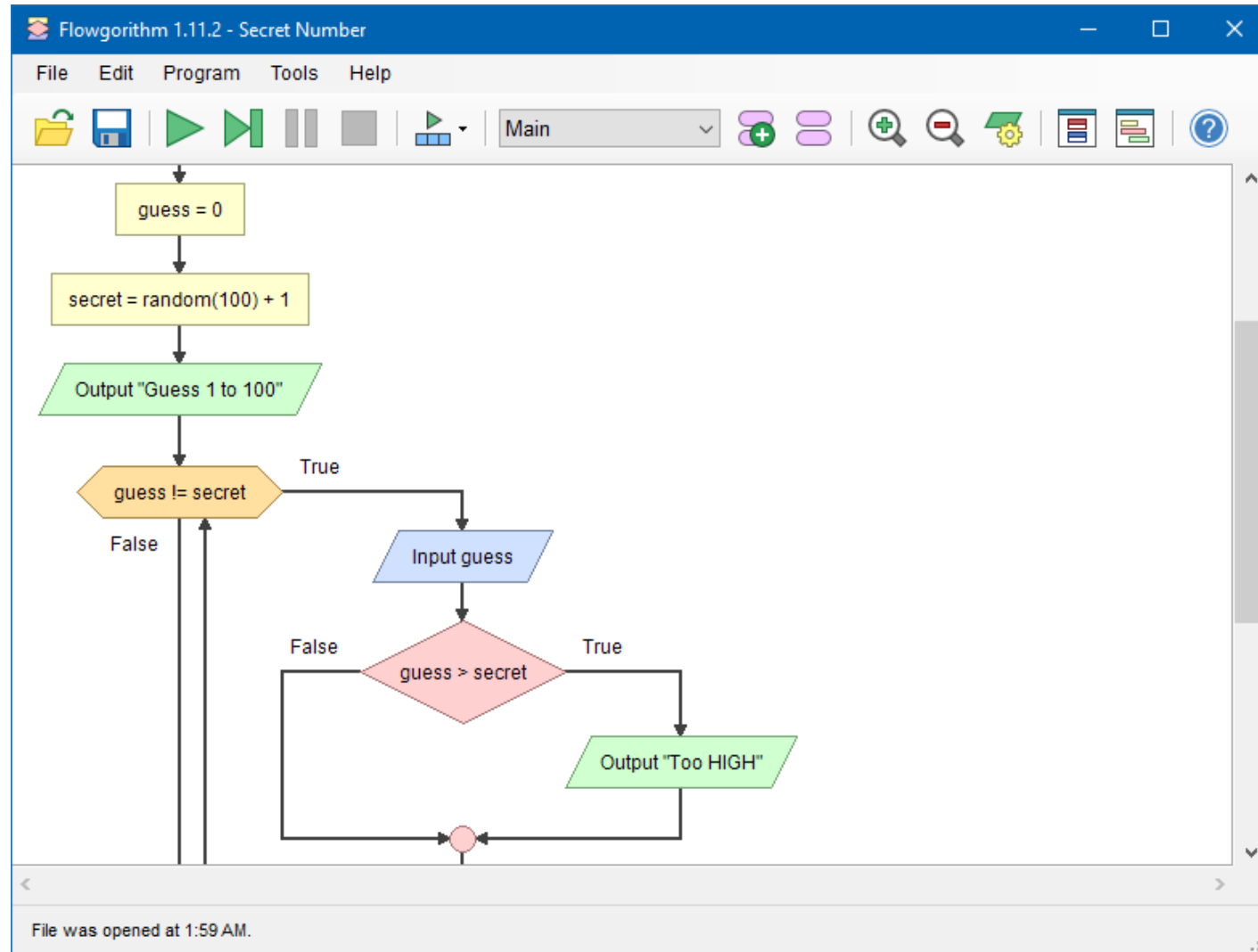
- **Nút đầu cuối** (terminal) hình ellipse
- **Nút nhập xuất** (input/output) hình bình hành
- **Nút thao tác** (activity) hình chữ nhật
- **Nút lựa chọn** (decision) hình thoi

Tìm hiểu thêm về lưu đồ!

- <https://en.wikipedia.org/wiki/Flowchart>



Lưu đồ có phải là một ngôn ngữ lập trình? (1)



Lưu đồ có phải là một ngôn ngữ lập trình? (2)

- Lưu đồ là một trong các phương tiện mô tả **trực quan** công việc
- Ngôn ngữ tự nhiên
- Mã giả
- Lưu đồ
- “Mã thật”

Khuôn chương trình “thực thi có lựa chọn” (1)

Tiếng Việt	Python
Nhập số thực x Nếu x không âm thì tính và xuất căn x	<pre>x = float(input("Enter a real number x: ")) if x >= 0: sqrt_x = x ** 0.5 print("sqrt(x) = %.6f" % sqrt_x) print("Done!")</pre>
Lưu đồ	C/C++
<pre>graph TD Start(()) --> Input[/Nhập x/] --> Decision{x ≥ 0?} Decision -- Đúng --> Process[r = √x] Process --> Output[/Xuất r/] --> End(()) Decision -- Sai --> End</pre>	<pre>#include <stdio.h> #include <math.h> int main() { double x, sqrt_x; printf("Enter a real number x: "); scanf("%lf", &x); if (x >= 0) { sqrt_x = sqrt(x); printf("sqrt(x) = %.6lf\n", sqrt_x); } printf("Done!\n"); }</pre>

Cấu trúc điều khiển là gì?

- **Cấu trúc điều khiển** (control structure) là các cấu trúc qui định dạng luồng thực thi
- Chúng được hiện thực (cài đặt) bằng các lệnh **điều khiển luồng thực thi** (controlling execution flow) trong các ngôn ngữ lập trình

Cấu trúc chọn là gì?

- **Cấu trúc chọn** (selection structure) cho phép lựa chọn thực thi hay không các lệnh nào đó dựa trên các **điều kiện** (condition)
- Chúng thường được hiện thực bằng các **lệnh chọn** (selection statement), còn gọi là **lệnh điều kiện** (conditional statement) hay **lệnh rẽ nhánh** (branch statement)
- Hai dạng lệnh chọn hay gặp nhất là “**if khuyết**” và “**if đủ**”

Lệnh “if khuyết” được viết thế nào?

- Python

```
if <điều kiện>:  
    <thân lệnh>
```

- C/C++

```
if (<điều kiện>)  
    <thân lệnh>
```

Lệnh “if khuyết” được thực thi thế nào?

- C/C++

```
if (<điều kiện>)  
    <thân lệnh>
```

- Nếu <điều kiện> đúng thì <thân lệnh> được thực thi; nếu sai thì thôi

Mô tả điều kiện của lệnh chọn bằng cách nào?

- Điều kiện của các lệnh chọn được mô tả bởi biểu thức luận lý
- **Biểu thức luận lý** (boolean expression) là biểu thức có **giá trị luận lý** (boolean value), hoặc **đúng** (true) hoặc **sai** (false)
- Biểu thức luận lý thông thường nhất là biểu thức so sánh số với các **toán tử so sánh** (comparison operator), còn gọi là **toán tử quan hệ** (relational operator)

Các toán tử so sánh nào hay được dùng?

- Python/C/C++ đều dùng chung các toán tử so sánh (số) là: $>$ (lớn hơn), $>=$ (lớn hơn hoặc bằng), $<$ (nhỏ hơn), $<=$ (nhỏ hơn hoặc bằng), $==$ (bằng), $!=$ (khác)
- Các toán tử gồm 2 kí hiệu phải được viết liền nhau
- Phân biệt toán tử so sánh bằng ($==$) với dấu/toán tử gán ($=$)
- Các toán tử so sánh có độ ưu tiên như nhau và thấp hơn các toán tử số học ($+$, $-$, ...)

Khối lệnh là gì?

- **Khối lệnh** (block) là dãy các lệnh được gom chung thành một “lô” thực thi: chúng cùng được thực thi tuần tự hoặc không thực thi
- Khối lệnh có vai trò như một lệnh
- Một lệnh là trường hợp đặc biệt của khối lệnh (khối chỉ có một lệnh)
- Khối lệnh lớn nhất là **khối chương trình** (program block)
- Các khối lệnh nhỏ hơn là các thân hàm (sẽ học)
- Các khối lệnh nhỏ hơn nữa là thân các “lệnh phức” như lệnh if

Khối lệnh được viết thế nào?

- Python:
 - các lệnh cùng khối được viết **thụt đầu dòng** (indent) cùng mức
 - các khối lệnh “phân cấp” được viết với các mức thụt đầu dòng khác nhau
- C/C++

```
{  
    <dãy các lệnh>  
}
```
- Các khối lệnh có quan hệ “phân cấp” hay “lồng chứa”

Khối lệnh được thực thi thế nào?

- C/C++

{

<lệnh 1>

<lệnh 2>

...

<lệnh n>

}

- Các lệnh con trong khối được thực thi tuần tự

Lưu ý về việc khai báo biến trong C/C++!

- Các biến khai báo trong một khối chỉ có “**phạm vi**” (scope) trong khối đó (và các khối “con cháu”), nghĩa là không thể dùng các biến ở “ngoài” khối khai báo

Khuôn chương trình “thực thi có lựa chọn” (2)

Mã giả	Python
<p>Nhập số nguyên x</p> $ x = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$ <p>Xuất x</p>	<pre>x = int(input("Enter an integer x: ")) if x >= 0: abs_x = x else: abs_x = -x print(" x = ", abs_x)</pre>
Lưu đồ	C/C++
<pre>graph TD Start([Nhập x]) --> Decision{x >= 0?} Decision -- Đúng --> Process1[x = x] Decision -- Sai --> Process2[x = -x] Process1 --> End([Xuất x]) Process2 --> End</pre>	<pre>#include <stdio.h> int main() { int x, abs_x; printf("Enter an integer x: "); scanf("%d", &x); if (x >= 0) abs_x = x; else abs_x = -x; printf(" x = %d\n", abs_x); }</pre>

Lệnh “if đủ” được viết thế nào?

- Python

```
if <điều kiện>:  
    <thân lệnh 1>  
else:  
    <thân lệnh 2>
```

- C/C++

```
if (<điều kiện>)  
    <thân lệnh 1>  
else  
    <thân lệnh 2>
```

Lệnh “if đủ” được thực thi thế nào?

- C/C++

```
if (<điều kiện>)  
    <thân lệnh 1>  
else  
    <thân lệnh 2>
```

- Nếu <điều kiện> đúng thì <thân lệnh 1> được thực thi; nếu sai thì <thân lệnh 2> được thực thi

Lệnh đơn khác lệnh phức thế nào?

- **Lệnh đơn** (simple statement) là các lệnh không chứa lệnh khác bên trong
 - lệnh nhập/xuất (thật ra là lời gọi các hàm tương ứng)
 - lệnh gán
 - ...
- **Lệnh phức** hay **lệnh ghép** (compound statement) là lệnh có chứa lệnh khác bên trong
 - khối lệnh
 - lệnh chọn
 - ...

Cộng điểm

- Làm bài tập 1.5.1 (Tài liệu C)
- Làm bài tập 1.5.2

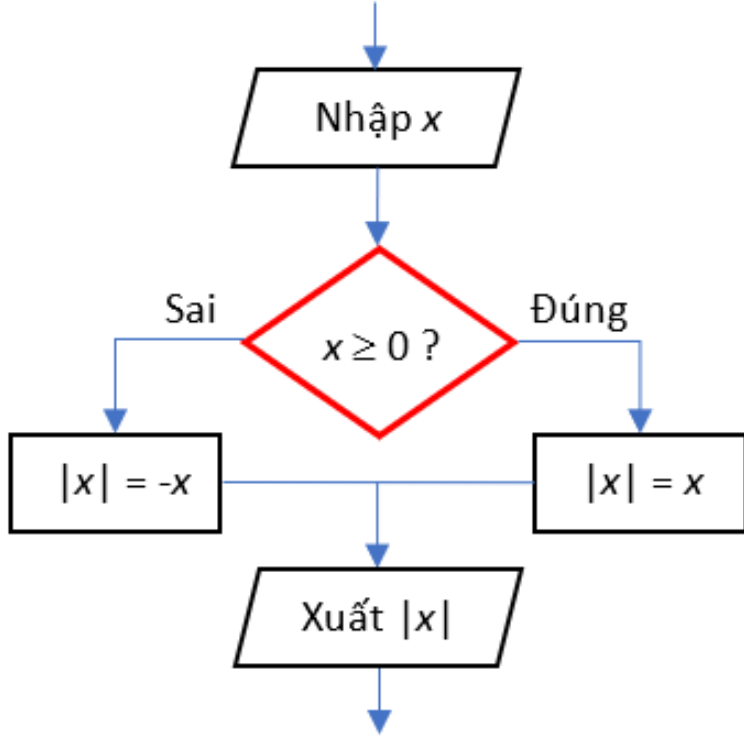
Giải lao!!!

- Giải lao đến 9h40

Lưu ý về giá trị luận lý và kiểu luận lý!

- Python có giá trị luận lý và kiểu luận lý: `True`, `False`, `bool`
- C++ có giá trị luận lý và kiểu luận lý: `true`, `false`, `bool`
- C không có giá trị luận lý và kiểu luận lý riêng (mà dùng kiểu số)
- Python/C/C++ đều dùng qui ước chung là:
 - Các giá trị số (nguyên, thực) khác 0 (0.0) được xem là đúng và 0 (0.0) được xem là sai
 - Giá trị luận lý đúng được xem là số nguyên 1 và giá trị luận lý sai được xem là số nguyên 0
- (Xem minh họa)

Tính trị tuyệt đối không dùng lệnh chọn!

Mã giả	Python
<p>Nhập số nguyên x</p> $ x = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$ <p>Xuất x</p>	<pre>x = int(input("Enter an integer x: ")) abs_x = (x if x >= 0 else -x) print(" x = ", abs_x)</pre>
Lưu đồ	C/C++
 <pre>graph TD Start([Nhập x]) --> Decision{x ≥ 0 ?} Decision -- Sai --> Process1[x = -x] Decision -- Đúng --> Process2[x = x] Process1 --> End[/Xuất x /] Process2 --> End</pre>	<pre>#include <stdio.h> int main() { int x, abs_x; printf("Enter an integer x: "); scanf("%d", &x); abs_x = (x >= 0 ? x : -x); printf(" x = %d\n", abs_x); }</pre>

Biểu thức điều kiện là gì?

- **Biểu thức điều kiện** (conditional expression) là biểu thức dùng **toán tử điều kiện** (conditional operator)
- Toán tử điều kiện là toán tử 3 ngôi giúp lựa chọn giá trị giữa 2 trường hợp tùy theo một điều kiện là đúng hay sai
- Python

**<A> if <Cond> else **

- C/C++

**<Cond> ? <A> : **

- Toán tử điều kiện thường có độ ưu tiên rất thấp (thấp hơn các toán tử số học, so sánh và logic)

Nên dùng biểu thức điều kiện khi nào?

- Biểu thức điều kiện được dùng khi lựa chọn các tính toán đơn giản
- Lệnh chọn được dùng khi lựa chọn các xử lý phức tạp
- Có thể viết biểu thức điều kiện lồng bằng cách dùng biểu thức điều kiện bên trong biểu thức điều kiện (**nhưng không nên lạm dụng vì khó đọc**)

Cộng điểm

- Làm lại bài tập 1.5.1 (Tài liệu C) dùng biểu thức điều kiện

Dùng lệnh nào khi có quá nhiều lựa chọn?

Mã giả	Python
Cho số nguyên x $\text{dấu của } x = \begin{cases} -, & x < 0 \\ 0, & x = 0 \\ +, & x > 0 \end{cases}$	<pre>x = int(input("Enter an integer x: ")) if x < 0: print("x is negative") else: if x == 0: print("x is zero") else: # x > 0 print("x is positive")</pre>
Lưu đồ	C/C++
<pre>graph TD Start([Nhập x]) --> D1{x < 0 ?} D1 -- Đúng --> O1[Xuất -] D1 -- Sai --> D2{x = 0 ?} D2 -- Đúng --> O2[Xuất 0] D2 -- Sai --> O3[Xuất +] O1 --> End([]) O2 --> End O3 --> End</pre>	<pre>#include <iostream> using namespace std; int main() { int x; cout << "Enter an integer x: "; cin >> x; if (x < 0) cout << "x is negative" << endl; else { if (x == 0) cout << "x is zero" << endl; else // x > 0 cout << "x is positive" << endl; } }</pre>

“if lồng” là gì?

- Chuyển lựa chọn 3 trường hợp thành 2 lựa chọn 2 trường hợp lồng nhau!
- Lệnh “if khuyết” và “if đủ” giúp ta hiện thực các cấu trúc chọn 1 và 2 lựa chọn
- Để hiện thực cấu trúc nhiều lựa chọn ta có thể vận dụng cấu trúc “chọn lồng” là **if lồng** (nested if)
 - lệnh if được dùng bên trong thân của lệnh if lớn hơn (ở phần if và/hoặc phần else)
- Số mức hay cấp lồng là tùy ý (**nhưng không nên lạm dụng**)

Một trường hợp hay gặp của “if lồng”

<p>Cho số nguyên x, dấu của x được xác định như sau:</p> <ul style="list-style-type: none">- $x < 0$: dấu -- $x = 0$: dấu 0- $x > 0$: dấu +	<pre>x = int(input("Enter an integer x: ")) if x < 0: print("x is negative") elif x == 0: print("x is zero") else: # x > 0 print("x is positive")</pre>
<p>Sơ đồ luồng (Flowchart)</p> <pre>graph TD Start([Nhập x]) --> D1{x < 0 ?} D1 -- Đúng --> O1[/Xuất -/] D1 -- Sai --> D2{x = 0 ?} D2 -- Đúng --> O2[/Xuất 0/] D2 -- Sai --> O3[/Xuất +/] O1 --> Exit(()) O2 --> Exit O3 --> Exit</pre>	<p>C/C++</p> <pre>#include <iostream> using namespace std; int main() { int x; cout << "Enter an integer x: "; cin >> x; if (x < 0) cout << "x is negative" << endl; else if (x == 0) cout << "x is zero" << endl; else // x > 0 cout << "x is positive" << endl; }</pre>

“if tầng” là gì?

- Một trường hợp đặc biệt của if lồng hay gộp là **if tầng** (cascading if) hay **if nối** (chained if), cho phép ta kiểm tra một dãy nhiều điều kiện theo nguyên tắc “đúng đầu tiên”
- Trong C/C++, if tầng thật ra là một trường hợp của if lồng được viết lại cho dễ đọc (if lồng cũng chỉ là vận dụng của lệnh if và khối lệnh)
- Trong Python, if tầng thực sự được hỗ trợ bằng các phần `elif` tùy chọn với số lượng tùy ý
- Có thể nối thêm nhiều phần `elif` (hay “else if”) để có nhiều tầng hơn

Cộng điểm

- Làm bài tập 1.5.3 (Tài liệu C)

Làm gì khi “if lồng” quá phức tạp?

- (Xem mã minh họa)

Toán tử logic là gì?

- Để mô tả các điều kiện phức tạp ta có thể dùng các **toán tử luận lý** (boolean operator) hay **toán tử logic** (logical operator)

<i>x</i>	<i>y</i>	<i>x and y</i>	<i>x or y</i>	<i>not x</i>
Sai	Sai	Sai	Sai	Đúng
Sai	Đúng	Sai	Đúng	Đúng
Đúng	Sai	Sai	Đúng	Sai
Đúng	Đúng	Đúng	Đúng	Sai

- Python: **and**, **or**, **not** C/C++: **&&**, **||**, **!**
- Các toán tử logic có độ ưu tiên cao hơn toán tử điều kiện nhưng thấp hơn các toán tử so sánh, trong đó độ ưu tiên giảm dần là not, and, or

Lượng giá tắt là gì?

- **Lượng giá tắt** (short-circuit evaluation) là cách kết thúc sớm việc lượng giá khi đã xác định được giá trị của biểu thức
- Các biểu thức $\langle A \rangle$ and $\langle B \rangle$, $\langle A \rangle$ or $\langle B \rangle$ thường được lượng giá tắt
- (Xem mã minh họa)

Cộng điểm

- Làm bài tập 2.4.2 (Tài liệu C)
- Làm bài tập 2.4.3