

TÓM LƯỢC BÀI GIẢNG NHẬP MÔN LẬP TRÌNH

(Vũ Quốc Hoàng, vqhoang@fit.hcmus.edu.vn, FIT-HCMUS, 2020)

BÀI 4 CẤU TRÚC CHỌN

Chủ đề

- Toán tử và biểu thức điều kiện
- Cấu trúc chọn và lệnh chọn
- Toán tử logic

Tài liệu

- [1] Vũ Quốc Hoàng, *Bí kíp luyện Lập trình C (Quyển 1)*, hBook, 2017.
- [2] Tony Gaddis, *Starting out with C++ From Control Structures through Objects*, Pearson, 8th edition, 2015.
- [3] Vũ Quốc Hoàng, *Bí kíp luyện Lập trình nhập môn với Python*, hBook, 2020.
- Đọc kĩ: Bài 2.4 [1]
 - Đọc thêm: Phần 4.5-4.9, 4.13, 4.15 [2]; Bài 6 [3]

Kiến thức

- **Biểu thức điều kiện** (conditional expression) là biểu thức dùng **toán tử điều kiện** (conditional operator). Đây thường là toán tử 3 ngôi giúp ta lựa chọn giá trị giữa 2 trường hợp tùy theo một điều kiện là đúng hay sai. Biểu thức điều kiện giúp mô tả các lựa chọn tính toán đơn giản hơn lệnh chọn.
- 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), trong đó, 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).
- Ta cũng 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).
- 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”.
- Để mô tả các điều kiện phức tạp (tạo nên từ các điều kiện đơn giản hơn) ta có thể dùng các **toán tử luận lý** (boolean operator), hay **toán tử logic** (logical operator), là phủ định (not), “và” (and), “hoặc” (or). Các toán tử này được dùng với ý nghĩa thông thường như mô tả trong bảng sau:

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

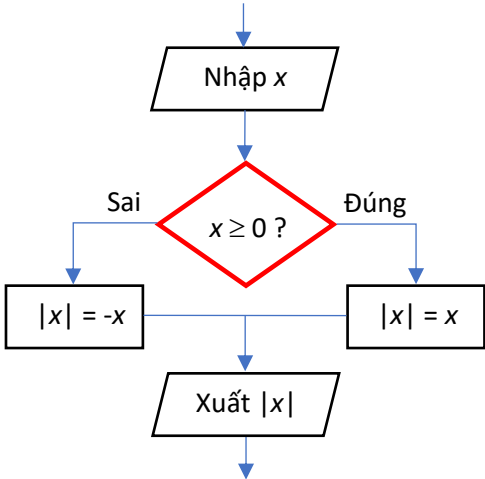
Để dễ nhớ: phủ định cho kết quả ngược lại, x and y chỉ đúng khi cả x và y đều đúng, x or y chỉ sai khi cả x và y đều sai.

Kĩ năng

- Vẽ được và phân tích được lưu đồ cho các việc đơn giản
- Cài đặt được chương trình từ lưu đồ và vẽ được lưu đồ từ mã nguồn chương trình
- Thành thạo việc dùng toán tử điều kiện, toán tử logic, if lồng, if tầng
- Viết được chương trình có sử dụng các cấu trúc chọn
- Ý thức và rèn luyện được việc viết chương trình đẹp, ngăn nắp, rõ ràng

Mã nguồn minh họa

Mã nguồn 1. (Biểu thức điều kiện)

Mã giả	Python
Nhập số nguyên x $ x = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$ Xuất $ x $	<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 A[/Nhập x/] --> B{x >= 0 ?} B -- Đúng --> C[x = x] B -- Sai --> D[x = -x] C --> E[/Xuất x /] D --> E </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>
Giải thích và bình luận thêm: <ul style="list-style-type: none"> - Cú pháp của biểu thức điều kiện trong Python là $\langle A \rangle \text{ if } \langle C \rangle \text{ else } \langle B \rangle$ của C/C++ là $\langle C \rangle ? \langle A \rangle : \langle B \rangle$ trong đó biểu thức A mô tả kết quả của biểu thức điều kiện khi điều kiện (biểu thức luận lý) C đúng còn biểu thức B mô tả kết quả khi C sai; cách viết của C/C++ "tự nhiên" hơn của Python - if else và ? : ở trên là các toán tử (3 ngôi) - Toán tử điều kiện thường có độ ưu tiên rất thấp (thấp hơn các toán tử số học, so sánh và logic), do đó, nên dùng cặp ngoặc tròn để bọc biểu thức điều kiện cho "chắc ăn" - Lưu đồ của mã nguồn này hoàn toàn giống lưu đồ của mã nguồn 3 trong Bài 2A; toán tử điều kiện thường giúp thay thế lệnh chọn trong trường hợp lựa chọn các tính toán đơn giản thay vì lựa chọn các xử lý phức tạp - Trường hợp nhiều hơn 2 lựa chọn thì làm sao? 	

Mã nguồn 2. (if lồng)

Mã giả	Python
<p>Cho số nguyên x</p> $\text{dấu của } x = \begin{cases} -, & x < 0 \\ 0, & x = 0 \\ +, & x > 0 \end{cases}$	<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 --> Exit([X]) O2 --> Exit O3 --> Exit </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>
<p>Giải thích và bình luận thêm:</p> <ul style="list-style-type: none"> - Ta chuyển lựa chọn 3 trường hợp thành 2 lựa chọn 2 trường hợp lồng nhau - Vì thân if là khối lệnh gồm dãy lệnh bất kì, có thể là lệnh if khác, nên cấu trúc if lồng không có gì mới về cú pháp mà chỉ là sự vận dụng if đủ (và if khuyết) - Cặp ngoặc nhọn trong phần else của lệnh if ngoài ở mã C/C++ trên có thể được bỏ đi vì khối lệnh này chỉ gồm 1 lệnh (là lệnh if đủ bên trong), tuy nhiên, ta nên giữ để mã nguồn trông rõ ràng và “cứng cáp” hơn - Ta cũng có thể dùng “biểu thức điều kiện lồng” để viết lại các mã nguồn trên đơn giản hơn 	

Mã nguồn 3. (if tầng)

Tiếng Việt	Python
<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>

Sơ đồ luồng (Flowchart)	C/C++
<pre> graph TD Start([Nhập x]) --> Cond1{x < 0 ?} Cond1 -- Đúng --> Out1[/Xuất -/] Cond1 -- Sai --> Cond2{x = 0 ?} Cond2 -- Đúng --> Out2[/Xuất 0/] Cond2 -- Sai --> Out3[/Xuất +/] Out1 --> Join(()) Out2 --> Join Out3 --> Join Join --> 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>
Giải thích và bình luận thêm: <ul style="list-style-type: none"> - Lưu đồ của mã nguồn này hoàn toàn giống lưu đồ của mã nguồn 2; thực chất if tầng chỉ là một trường hợp đặc biệt (hay gặp) của if lồng - Mã nguồn C/C++ này là mã nguồn C/C++ ở mã nguồn 2 sau khi bỏ đi cặp ngoặc nhọn trong phần else của lệnh if ngoài và định dạng lại; nhớ rằng việc định dạng không quan trọng với C/C++ mà chỉ để mã dễ đọc; như vậy không có cái gọi là if tầng trong C/C++, nó thật ra là một trường hợp của if lồng được viết lại cho dễ đọc; (cũng nhớ là không thật sự có cái gọi là if lồng, nó chỉ là vận dụng của if và khối lệnh) - Python thì khác, Python thực sự (về cú pháp hay ngôn ngữ) có hỗ trợ if tầng bằng các phần elif tùy chọn với số lượng tùy ý - Như tên gọi, ta có thể nối thêm nhiều phần elif (hay "else if") để có nhiều tầng hơn 	

Mã nguồn 4. (Điều kiện phức tạp)

Python (if tầng và if lồng)	Python (Điều kiện phức tạp)
<pre> a = int(input("Enter an integer a: ")) b = int(input("Enter an integer b: ")) c = int(input("Enter an integer c: ")) if a <= b: if b <= c: # a b c m = b elif a <= c: # a c b m = c else: # c a b m = a else: if a <= c: # b a c m = a elif b <= c: # b c a m = c else: # c b a m = b </pre>	<pre> a = int(input("Enter an integer a: ")) b = int(input("Enter an integer b: ")) c = int(input("Enter an integer c: ")) if (b <= a <= c) or (c <= a <= b): m = a if (a <= b <= c) or (c <= b <= a): m = b if (a <= c <= b) or (b <= c <= a): m = c print("The middle is", m) </pre>

print("The middle is", m)	
C/C++ (if lồng)	C/C++ (Điều kiện phức tạp)
<pre> #include <stdio.h> int main() { int a, b, c, m; printf("Enter 3 integers: "); scanf("%d %d %d", &a, &b, &c); if (a <= b) { if (b <= c) // a b c m = b; else if (a <= c) // a c b m = c; else // c a b m = a; } else { if (a <= c) // b a c m = a; else if (b <= c) // b c a m = c; else // c b a m = b; } printf("The middle is %d\n", m); } </pre>	<pre> #include <stdio.h> int main() { int a, b, c, m; printf("Enter 3 integers: "); scanf("%d %d %d", &a, &b, &c); if ((b <= a && a <= c) (c <= a && a <= b)) m = a; if ((a <= b && b <= c) (c <= b && b <= a)) m = b; if ((a <= c && c <= b) (b <= c && c <= a)) m = c; printf("The middle is %d\n", m); } </pre>
<p>Giải thích và bình luận thêm:</p> <ul style="list-style-type: none"> - Ta có thể mô tả các điều kiện phức tạp bằng các toán tử logic thích hợp trên các điều kiện đơn giản hơn; toán tử phủ định được kí hiệu trong Python là not, trong C/C++ là !; toán tử “và” được kí hiệu trong Python là and, trong C/C++ là &&; toán tử “hoặc” được kí hiệu trong Python là or, trong C/C++ là - Toán tử phủ định có độ ưu tiên cao hơn toán tử “và”; toán tử “và” có độ ưu tiên cao hơn toán tử “hoặc” - Đôi khi việc dùng toán tử logic để mô tả các điều kiện phức tạp sẽ cho mã nguồn đơn giản hơn; đôi khi việc dùng if lồng (và/hoặc if tầng) lại tự nhiên hơn; việc vận dụng các cấu trúc chọn khác nhau cho các công việc phù hợp là một nghệ thuật - Có một cách định dạng khối “hiện đại” hơn như minh họa trên trong C/C++, so với cách định dạng “cổ điển” hơn ở các mã nguồn trước - Cách viết $x \leq y \leq z$ trong Toán có ngữ nghĩa là $x \leq y$ và $y \leq z$; C/C++ yêu cầu ta phải viết rõ ra; Python thì cho phép viết “nối” - Cần tìm số nằm giữa trong 5 số thì làm sao? 	

Bài tập

1. Làm các bài tập sau theo C, C++: 1.5.2-1.5.4, 2.3.4, 2.4.1-2.4.6 [1].
2. Làm thêm các bài tập sau bằng Python: Bài 6 [3].