TÓM LƯỢC BÀI GIẢNG
NHẬP MÔN LẬP TRÌNH

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

BÀI 3
LUỒNG THỰC THI VÀ LƯU ĐỒ

Chủ đề

- Luồng thực thi
- Lưu đồ
- Cấu trúc điều khiển
- Cấu trúc chọn và lệnh chọn
- Khối lệnh
- Giá trị luận lý và toán tử so sánh

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 1.5 (trang 29-31), 2.3, 2.4 (trang 93-94) [1]
- Đọc thêm: Phần 2.10, 2.13, 4.1-4.4 [2]; Bài 4, 5 [3]

Kiến thức

- Thứ tự thực thi các lệnh của một chương trình được gọi là **luồng thực thi** (execution flow, control flow). Một quá trình thực thi cụ thể, tức diễn tiến cùng với trạng thái và kết quả thực thi, của một chương trình được gọi là một **tiến trình** (process). 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.
- **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 hay dùng là: nút đầu cuối hình ellipse, nút nhập xuất hình bình hành, nút thao tác hình chữ nhật và nút lựa chọn hình thoi. Nút lựa chọn kiểm tra một điều kiện và có 2 cạnh ra tùy theo điều kiện là đúng hay sai.
- 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 ở dưới đó). Thực thi tuần tự tương ứng với lưu đồ không có nút lựa chọn và không có **chu trình** (cycle).
- **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** (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). 2 dạng lệnh chọn hay gặp nhất là **if đủ** và **if khuyết**.
- **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). Biểu thức luận lý thường được dùng để mô tả điều kiện của các lệnh chọn.
- **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. Về logic, khối lệnh có vai trò như một lệnh hay 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).
- **Lệnh đơn** (simple statement) là các lệnh không có chứa lệnh khác bên trong như 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 như khối lệnh, lệnh if đủ, if khuyết, ...

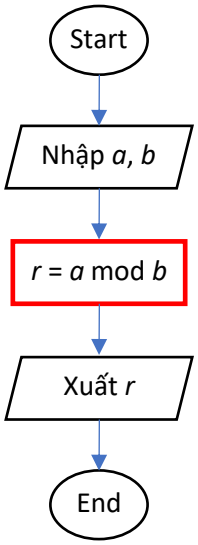
Kĩ năng

- 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 so sánh, lệnh if đủ, if khuyết và khối lệnh
- Viết được chương trình có sử dụng các cấu trúc chọn đơn giản
- Ý thức và rèn luyện được việc viết chương trình đẹp, ngăn nắp, rõ ràng

Mã nguồn minh họa

Mã nguồn 1. (Thực thi tuần tự)

Lưu đồ (Flowchart)



Python

```
a = int(input("Enter an integer a: "))
b = int(input("Enter an integer b: "))
print("a mod b = ", a % b)
print("Done!")
```

C/C++

```
#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");
}
```

Giải thích và bình luận thêm:

- Trong lưu đồ cũng như mã giả, ta không bận tâm các chi tiết (như việc khai báo biến hay việc dùng đúng kí hiệu toán tử của ngôn ngữ lập trình nào đó); mod (viết tắt của modulo) là từ hay được dùng để kí hiệu cho phép chia lấy dư
- Việc thực thi tuần tự có thể "dừng sớm" khi gặp lỗi thực thi
- Có thể xem lưu đồ là biểu diễn trực quan của chương trình: cài đặt chương trình từ lưu đồ thường dễ hơn là từ mã giả hay từ đầu, đọc lưu đồ dễ hình dung luồng thực thi hơn là đọc mã nguồn
- Có thể có vấn đề gì với nút nhập không? (Việc nhập có khả năng phát sinh lỗi thực thi không?)

Mã nguồn 2. (if khuyết)

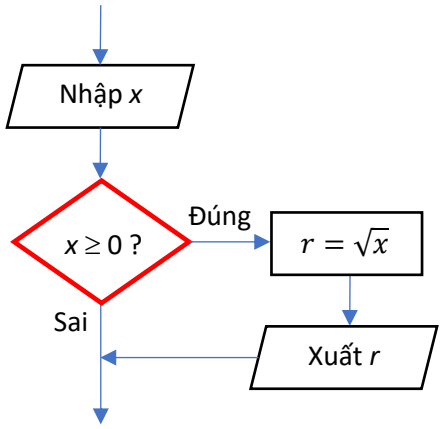
Tiếng Việt

Nhập số thực *x*
Nếu *x* không âm
    thì tính và xuất căn *x*

Python

```
x = float(input("Enter a real number x: "))
if x >= 0:
    sqrt_x = x ** 0.5
    print("sqrt(x) = %.6f" % sqrt_x)
print("Done!")
```

Lưu đồ



C/C++

```
#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");
}
```

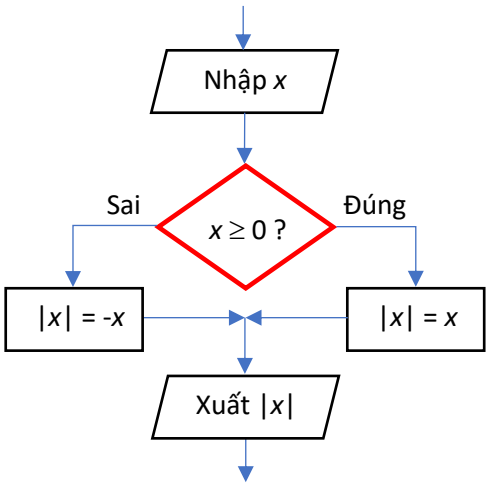
Giải thích và bình luận thêm:

- 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 gán (=)
- Các toán tử so sánh có độ ưu tiên thấp hơn các toán tử số học (+, -, ...)
- Python dùng định dạng để xác định khối lệnh: các lệnh cùng một khối phải cùng mức thụt đầu dòng; C/C++ dùng các dấu câu: các lệnh cùng một khối để trong cặp ngoặc nhọn {...}
- Các khối lệnh có quan hệ "lồng chứa": một khối lệnh được chứa hẳn trong khối lớn hơn (khối "con" chứa trong khối "cha") hoặc "rời nhau"; khối lệnh lớn nhất (khối "ngoài cùng") được gọi là khối chương trình
- Với C/C++, các biến khai báo trong một khối chỉ có "phạm vi" trong khối đó (và các khối "con cháu"), nghĩa là không thể dùng các biến ở "ngoài" khối khai báo
- Gọi là if khuyết vì nó thiếu "else"
- Trường hợp *x* nhỏ hơn 0 thì sao?
- (Python cho phép làm việc với số phức; C/C++ không hỗ trợ "trực tiếp" số phức)

Mã nguồn 3. (if đủ)

<table>
<tr><th>Mã giả</th><th>Python</th></tr>
<tr><td>Nhập số nguyên x

$$|x| = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$$

Xuất $|x|$</td><td><pre>x = int(input("Enter an integer x: "))
if x >= 0:
    abs_x = x
else:
    abs_x = -x
print("|x| = ", abs_x)</pre></td></tr>
<tr><th>Lưu đồ</th><th>C/C++</th></tr>
<tr><td> 
</td><td><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></td></tr>
<tr><td colspan="2">Giải thích và bình luận thêm:
- Trong C/C++, khối chỉ có một lệnh thì không cần cặp ngoặc nhọn (đúng ra là khối lệnh được xem là một lệnh (lệnh kép))
- Gọi là if đủ vì nó có phần else
- Khối lệnh của phần if, phần else thường được gọi tương ứng là khối if, khối else (if block, else block) hoặc thân if, thân else (if body, else body)
- Trong C/C++, mình dấu chấm phẩy (<code>;</code>) đã được xem là lệnh, lệnh rỗng (null statement), do đó cần cẩn thận để không dùng dấu chấm phẩy sau if (<code>if(...);</code>) hay else (<code>else;</code>) vì việc này hợp lệ (không bị báo lỗi cú pháp) nhưng thường do nhầm lẫn hơn là dùng có chủ ý (ta sẽ thấy cách dùng có chủ ý sau)</td></tr>
</table>

Bài tập

1. Làm các bài tập sau theo C, C++: 1.5.1, 2.3.2 [1]
2. Làm thêm các bài tập sau bằng Python: Bài 4, 5 [3].