TÓM LƯỢC BÀI GIẢNG
NHẬP MÔN LẬP TRÌNH

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

BÀI 5A
CẤU TRÚC LẶP 1

Chủ đề

- Lưu đồ
- Cấu trúc lặp và lệnh lặp

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, 2.5 [1]
- Đọc thêm: Phần 5.2-5.6 [2]; Bài 7 [3]

Kiến thức

- Khả năng làm "lặp lại nhiều lần một việc đơn giản" là sở trường và sức mạnh của máy tính.
- **Cấu trúc lặp** (iteration structure, looping structure) mô tả việc lặp lại các công việc theo cách nào đó. Các cấu trúc lặp hay gặp là "lặp một số lần xác định", "lặp với biến lặp lần lượt nhận dãy các giá trị", "lặp trong khi", "lặp cho đến khi".
- Các ngôn ngữ lập trình cung cấp các **lệnh lặp** (iteration statement, looping statement) khác nhau để hiện thực các cấu trúc lặp phổ biến trên. Các lệnh lặp trong Python là `for`, `while`; các lệnh lặp trong C/C++ là `for`, `while`, do-`while`.
- **Thân lặp** (looping body) của các lệnh lặp (tức là công việc được lặp lại) cũng là khối lệnh và **điều kiện lặp** (looping condition) trong một số cấu trúc lặp cũng là biểu thức luận lý như thân và điều kiện của các lệnh chọn.

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 các lệnh lặp `for`, `while`, do-`while`
- Viết được chương trình có sử dụng các cấu trúc lặp
- Ý thức và rèn luyện được việc viết chương trình đẹp, ngăn nắp, rõ ràng

Mã nguồn minh họa

Mã nguồn 1. (Lặp số lần xác định)

Ngôn ngữ tự nhiên

Nói 100 lần câu sau đây: "I love you!".

Python

```
for i in range(100):
    print("I \u2764 you!")
```

C/C++

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

int main()
{
    for(int i = 1; i <= 100; i++)
        cout << "I love you!" << endl;
}
```

Giải thích và bình luận thêm:

- Lặp lại số lần xác định là dạng lặp cơ bản nhất; số lần lặp có thể xác định trước (là hằng số như 100 ở trên) hay xác định lúc chạy (là kết quả của một biểu thức nguyên như giá trị nhập từ người dùng)
- Lệnh được lặp lại gọi là **thân lặp** (loop body), là 1 khối lệnh; (nhớ lại, trong C/C++, khối lệnh gồm chỉ 1 lệnh có thể không cần cặp ngoặc nhọn { })
- Python hỗ trợ việc xuất ra nhiều kí hiệu "lạ" bao gồm các kí tự tiếng Việt (và nhiều kí tự khác) nếu biết **mã Unicode** (Unicode code point) của nó; mã Unicode của kí hiệu "trái tim" là U+2764 (viết trong hằng chuỗi là \u2764)
- `for`, `in`, `int` là từ khóa; `range` là hàm dựng sẵn
- Nếu hằng số 100 ở trên được thay thành 0 hay một số nguyên âm thì sao?
- Nếu hằng số 100 được thay thành một giá trị thực như 10.5 thì sao?

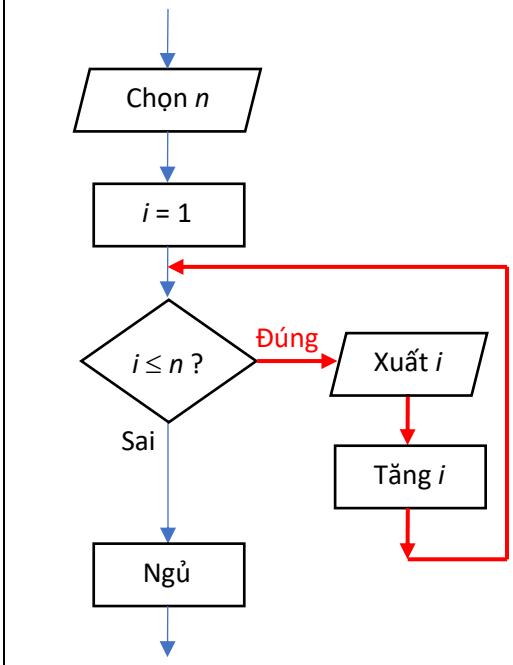
Mã nguồn 2. (Lặp theo biến lặp)

Mã giả

Lặp lại việc sau đây với *i* lần lượt nhận các giá trị 1, 2, 3, ..., *n*:

- Đếm con cừu *i*

Lưu đồ



Python

```
import time

n = int(input("Đếm tới mấy? "))
for i in range(1, n + 1):
    print(i, "con cừu"); time.sleep(1.0)
    print(i, "con cừu"); time.sleep(1.0)
print("Ngủ đi!")
```

C/C++

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

int main()
{
    int n;

    printf("Dem toi may? ");
    cin >> n;

    for(int i = 1; i <= n; i++)
    {
        cout << i << " con cuu" << endl;
        cout << i << " con cuu" << endl;
    }
    cout << "Ngu di!" << endl;
}
```

Giải thích và bình luận thêm:

- `i` ở trên là biến và được gọi là **biến lặp** (loop variable); ta có thể đặt tên khác nếu muốn nhưng thường các biến lặp được đặt tên là `i`, `j`, k, ...
- Lệnh `for` điều khiển việc cho biến lặp lần lượt nhận các giá trị trong dãy giá trị nào đó và thực hiện thân lặp tương ứng mỗi lần; trong lần lặp tương ứng, ta có thể truy xuất giá trị mà biến lặp được nhận bằng biểu thức có chứa biến lặp (như thông thường)
- Trong Python, ta có thể cho chương trình tạm "nghỉ" (dừng) một thời gian nào đó bằng cách dùng hàm `sleep` của module `time` (đối số là thời gian dừng tính theo giây)
- Nhớ lại, Python dùng dấu chấm phẩy (`;`) để phân cách các lệnh khi viết trên cùng một dòng
- Ở trên, với mỗi *i*, ta đã "đếm" 2 lần "*i* con cừu", nếu muốn đếm 10 lần (với mỗi *i*) thì sao?
- Có thể dùng biến lặp thực không?

Mã nguồn 3. (Điều khiển chi tiết biến lặp)

Python (cách 1)

```
n = int(input("n = ? "))
for i in range(1, n + 1, 2):
    print(i)
```

Python (cách 2)

```
n = int(input("n = ? "))
for i in range(1, n + 1):
    if i % 2 == 1: # i lẻ
        print(i)
```

C/C++ (cách 1)

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

int main()
{
    int n;

    printf("n = ? ");
    cin >> n;

    for(int i = 1; i <= n; i += 2)
        cout << i << endl;
}
```

C/C++ (cách 2)

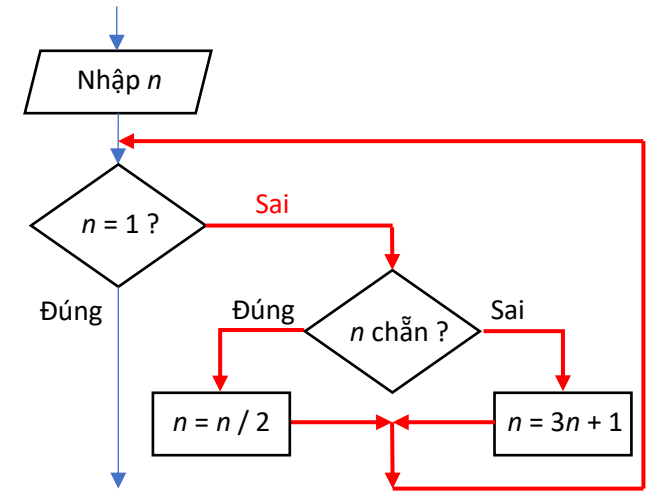
```
#include <iostream>
using namespace std;

int main()
{
    int n;

    printf("n = ? ");
    cin >> n;

    for(int i = 1; i <= n; i++)
        if(i % 2 == 1) // i le
            cout << i << endl;
}
```

Giải thích và bình luận thêm:

- Trong Python, hàm `range` có thể nhận 3 giá trị là start (giá trị bắt đầu), stop (giá trị kết thúc, không bao gồm) và step (mức tăng thêm) giúp điều khiển chi tiết hơn biến lặp; mặc định, start nhận giá trị 0 và step nhận giá trị 1
- Lệnh `for` của C/C++ thì mạnh mẽ và linh hoạt hơn (`for` của Python) với cấu trúc: `for(<Start>; <Stop>; <Step>)`, trong đó `<Start>`, `<Step>` là các lệnh (còn `<Stop>` là biểu thức luận lý), các thành phần này cũng có thể để trống
- Một cách khác để điều khiển chi tiết lệnh lặp là "lặp đơn giản với lệnh chọn bên trong", tức là dùng cấu trúc lặp đơn giản để lặp qua các giá trị và dùng lệnh chọn trong thân lặp để chọn ra các giá trị cần
- Ở trên ta đã chọn ra các số lẻ theo thứ tự tăng dần; nếu muốn theo thứ tự giảm dần thì sao?
- start, step, stop có thể nhận giá trị thực không?

Mã nguồn 4. (Lặp linh hoạt với `while`, do-`while`)

Lưu đồ



Mã giả

Cho *n* là một số nguyên dương, lặp lại quá trình sau đây cho đến khi *n* là 1 thì dừng:

- Nếu *n* chẵn: *n* = *n* / 2,
- Nếu *n* lẻ: *n* = 3*n* + 1.

Python (while)

```
n = int(input("Nhập n > 1: "))

while n != 1:
    print(n, end=" ")
    if n % 2 == 0: # n chẵn
        n = n // 2
    else:          # n lẻ
        n = 3*n + 1
print(n, end=" ")  # n là 1
```

C/C++ (do-while)

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

int main()
{
    int n;

    printf("Nhap n > 1: ");
    cin >> n;

    do
    {
        cout << n << " ";
        if(n % 2 == 0) // n chan
            n = n / 2;
        else           // n le
            n = 3 * n + 1;
    } while(n != 1);
    cout << n << " ";  // n la 1
}
```

C/C++ (while)

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

int main()
{
    int n;

    printf("Nhap n > 1: ");
    cin >> n;

    while(n != 1)
    {
        cout << n << " ";
        if(n % 2 == 0) // n chan
            n = n / 2;
        else           // n le
            n = 3 * n + 1;
    }
    cout << n << " ";  // n la 1
}
```

Giải thích và bình luận thêm:

- `while` là lệnh lặp có cấu trúc đơn giản nhất nhưng linh hoạt nhất, thường được dùng khi rất khó điều khiển biến lặp (hoặc "không có biến lặp" rõ ràng, hoặc "nhiều biến lặp", ...); trường hợp lặp số lần xác định hoặc biến lặp đơn giản thì nên dùng lệnh `for`
- **Điều kiện lặp** (looping condition) của `while` cũng là biểu thức luận lý như điều kiện chọn của lệnh `if`
- C/C++ phân biệt 2 dạng lặp là WHILE-DO (kiểm tra điều kiện lặp trước khi thực thi thân lặp) với lệnh `while` và DO-WHILE (thực thi thân lặp trước kiểm tra điều kiện lặp) với lệnh do-`while`; nên dùng do-`while` khi thân lệnh được lặp lại ít nhất 1 lần
- Tất cả các cấu trúc lặp đều có thể cài đặt chỉ với lệnh `while` (tương tự việc tất cả các cấu trúc chọn có thể cài đặt chỉ với lệnh `if`) nhưng nên chọn cách tự nhiên nhất, phù hợp nhất

- Nếu nhập $n = 1$ thì sao?
- Có số (nguyên dương) n nào để quá trình lặp trên là vô hạn không? (**nghi vấn $3n + 1$**)

Bài tập

1. Làm các bài tập sau theo C, C++: 1.5.5-1.5.8 [1].
2. Làm thêm các bài tập sau bằng Python: Bài 7 [3].