TÓM LƯỢC BÀI GIẢNG
NHẬP MÔN LẬP TRÌNH

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

BÀI 6A
HÀM 1

Chủ đề

- Định nghĩa hàm và gọi hàm
- Tham số và đối số
- Đối số mặc định
- Cách truyền đối số cho tham số

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 1.4, Bài 3.3 [1]
- Đọc thêm: Phần 6.1-6.4, 6.7-6.9, 6.12 [2]; Bài 8 [3]

Kiến thức

- **Hàm** (function) là một khối lệnh được đặt tên và tham số hóa. Hàm là công cụ quan trọng nhất của lập trình.
- Hàm được mô tả qua **định nghĩa hàm** (function definition) gồm **tên hàm** (function name), các **tham số** (parameter) và **thân hàm** (function body), chính là khối lệnh của hàm. Một số ngôn ngữ yêu cầu phải cho biết kiểu của các tham số và kiểu của **giá trị trả về** (return value) của hàm. Khai báo gồm tên hàm và các tham số (cùng với kiểu các tham số và kiểu giá trị trả về) được gọi là **nguyên mẫu** (prototype) của hàm. Các tham số còn được gọi là **đầu vào** (input) và giá trị trả về còn được gọi là **đầu ra** (output) của hàm.
- Hàm được dùng (tức là gọi chạy hay thực thi thân hàm) qua **lời gọi hàm** (function call) gồm tên hàm xác định hàm được gọi và các **đối số** (argument) xác định các giá trị truyền (gán) cho các tham số tương ứng trước khi thực thi thân hàm. Lời gọi hàm là một biểu thức mà giá trị chính là giá trị trả về của hàm sau khi hàm thực thi xong (nếu có).
- Các hàm có trả về giá trị thường được gọi là hàm tính toán còn các hàm không trả về giá trị thường được gọi là **thủ tục** (procedure).
- **Lệnh return** (return statement) yêu cầu hàm kết thúc thực thi và trả về giá trị tương ứng. Hàm cũng có thể kết thúc "tự nhiên" khi thực thi xong thân hàm (là khối lệnh).
- Có 2 cách cơ bản để **truyền (gán) giá trị của đối số cho tham số** (argument passing) là: (1) **theo vị trí** (passing argument by position), dựa trên thứ tự của đối số tương ứng với thứ tự của tham số;

(2) **theo tên** (passing argument by name), dựa trên tên của tham số đi kèm với đối số. Cách đầu tiện lợi hơn nhưng cách sau rõ ràng hơn.

- Các tham số cũng có thể được khai báo nhận **đối số mặc định** (default argument) là giá trị mà tham số sẽ nhận nếu lời gọi hàm không cung cấp đối số tương ứng. Do đó, các tham số này còn được gọi là **tham số tùy chọn** (optional parameter).
- Hầu hết các ngôn ngữ lập trình đều cung cấp rất nhiều hàm được định nghĩa sẵn trong các module, thư viện.

Kĩ năng

- Phân biệt được định nghĩa hàm với gọi hàm, tham số với đối số
- Viết được các hàm đơn giản
- Vận dụng được hàm để giải các bài toán
- Dùng được các hàm định nghĩa sẵn trong các module, thư viện
- Ý thức và rèn luyện được việc viết chương trình đẹp, ngăn nắp, rõ ràng

Mã nguồn minh họa

Mã nguồn 1. (Hàm tính toán)

Python

```python
import math


def mysqrt(x):
    y = x
    for i in range(100):
        y = y/2 + x/(2*y)
    return y



print(mysqrt(2))
print(mysqrt(4))



x = float(input("Enter x > 0: "))
print(mysqrt(x))
print(math.sqrt(x))
```

C/C++

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

double mysqrt(double x) {
    double y = x;
    for (int i = 1; i <= 100; i++)
        y = y/2 + x/(2*y);
    return y;
}

int main() {
    cout << mysqrt(2) << endl;
    cout << mysqrt(4) << endl;

    double x;
    cout << "Enter x > 0: ";
    cin >> x;
    cout << mysqrt(x) << endl;
    cout << sqrt(x) << endl;
}
```

Giải thích và bình luận thêm:

- Hàm giúp không phải "chép-sửa" một khối lệnh nhiều chỗ; định nghĩa một lần (một chỗ) và dùng nhiều lần (nhiều chỗ)
- Trong C/C++, ta phải khai báo kiểu của các tham số và kiểu của giá trị trả về
- Các hàm phải được định nghĩa trước khi dùng

- Để dễ phân biệt, tham số (trong định nghĩa hàm) được gọi là **tham số/đối số hình thức** (formal parameter/argument) còn đối số (trong lời gọi hàm) được gọi là **tham số/đối số thực tế** (actual parameter/argument)
- Tên hàm và tham số là các định danh nên được đặt theo qui tắc như tên biến; tên hàm nên dùng động từ; tên tham số nên dùng danh từ như tên biến
- Cặp ngoặc tròn phải có trong định nghĩa hàm và lời gọi hàm dù hàm không có tham số; trường hợp nhiều tham số thì dùng dấu phẩy phân cách (cũng vậy với đối số)
- Trong lệnh `return`, sau từ khóa `return` là một biểu thức, xác định kết quả trả về của hàm
- Lời gọi hàm là một biểu thức nên nó có thể được dùng ở mọi nơi chấp nhận một biểu thức
- Trong C/C++, chương trình được để trong hàm đặc biệt, có tên qui ước là `main`
- Cách tính căn bằng lặp ở trên được gọi là **phương pháp Newton** (Newton's method)
- Tham số `x` của hàm tính căn `mysqrt` chỉ nhận các giá trị dương; điều gì xảy ra nếu gọi `mysqrt` với đối số là số âm?

Mã nguồn 2. (Hàm không trả về giá trị - thủ tục)

Python

```python
def hello(name):
    hello_string = "Hi " + name
    length = len(hello_string)

    print("=" * (length + 4))

    print("| " + hello_string + " |")

    print("=" * (length + 4))



hello("Python")
hello("Guido van Rossum")



a_name = input("What's your name? ")
hello(a_name)
```

C/C++

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

void hello(string name) {
    string hello_string = "Hi " + name;
    int length = hello_string.length();
    for (int i = 1; i <= length + 4; i++)
        cout << "=";
    cout << endl;
    cout << "| " << hello_string << " |\n";
    for (int i = 1; i <= length + 4; i++)
        cout << "=";
    cout << endl;
}

int main() {
    hello("Python");
    hello("Guido van Rossum");

    string a_name;
    cout << "What's your name? ";
    getline(cin, a_name);
    hello(a_name);
}
```

Giải thích và bình luận thêm:

- Trong C/C++, từ khóa `void` được dùng để mô tả rằng hàm không trả về giá trị; khi đó lời gọi hàm tương ứng là một **biểu thức không giá trị** (void expression)
- Hàm không trả về giá trị thường được gọi là thủ tục và thường mô tả "làm gì đó" (chứ không phải "tính gì đó" như các hàm trả về giá trị)
- Thân hàm là một khối lệnh và có thể chứa mọi dạng lệnh; cũng vậy, hàm có thể nhận, xử lý và trả về mọi loại dữ liệu

- Lệnh `return` không đối số (tức là chỉ có từ khóa `return` mà không có biểu thức theo sau) có thể được dùng để kết thúc "sớm" hàm hoặc hàm kết thúc tự nhiên khi thực thi xong thân hàm

Mã nguồn 3. (Đối số mặc định)

Python

```python
def power(x=1, n=0):
    y = 1
    for i in range(n):
        y *= x
    return y


print(power(2, 64))
print(power(2, 0))
print(power(2))
print(power())
```

C++

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

double power(double x = 1, int n = 0) {
    double y = 1;
    for (int i = 1; i <= n; i++)
        y *= x;
    return y;
}

int main() {
    cout << power(2, 64) << endl;
    cout << power(2, 0) << endl;
    cout << power(2) << endl;
    cout << power() << endl;
}
```

Giải thích và bình luận thêm:

- Đa số các ngôn ngữ lập trình hỗ trợ đối số mặc định (Python, C++ có hỗ trợ, C không hỗ trợ)
- Để an toàn, các đối số mặc định nên là các "biểu thức hằng"
- Chỉ nên dùng đối số mặc định khi nó có ý nghĩa (tức là có một giá trị rất hay dùng hoặc ngầm định dùng cho tham số)

Mã nguồn 4. (Cách truyền tham số cho đối số)

Python

```python
for power(x=1, n=0):
    y = 1
    for i in range(n):
        y *= x
    return y

print("2^64 = ", power(2, 64))
print("64^2 = ", power(64, 2))
print("2^64 = ", power(n=64, x=2))
print("2^64 = ", power(x=2, n=64))
print("2^64 = ", power(2, n=64))
print("64^0 = ", power(64))
print("1^64 = ", power(n=64))
# print("2^64 = ", power(x = 2, 64)) # Syntax error
```

Giải thích và bình luận thêm:

- Hầu hết các ngôn ngữ lập trình đều hỗ trợ cách truyền đối số cho tham số theo vị trí
- Một số ngôn ngữ hỗ trợ cách truyền đối số cho tham số theo tên (Python hỗ trợ, C/C++ không hỗ trợ)

- Nên chọn thứ tự và tên các tham số một cách hợp lý

Bài tập

1. Làm các bài tập sau theo C, C++: 1.4.4, 3.3.1-3.3.8 [1].
2. Làm thêm các bài tập sau bằng Python: Bài 8 [3].