VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Kỹ Thuật Lập Trình (Cơ bản và nâng cao C++)

KTLT1 - HK242

TASK 3 FUNCTION

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 2/2025

Mục lục

1 Định nghĩa hàm

Hàm (function) trong C++ là một khối mã thực hiện một nhiệm vụ cụ thể. Việc sử dụng hàm giúp chương trình trở nên dễ đọc, dễ bảo trì và có thể tái sử dụng.

Cú pháp chung của một hàm:

```
return_type function_name(parameters) {
    // Khối lệnh
    return value;
}
```

Ví dụ:

```
int sum(int a, int b) {
    return a + b;
}
```

2 Cấu trúc một hàm

Một hàm trong C++ thường bao gồm các thành phần sau:

1. **Kiểu trả về (Return type)**: Xác định kiểu dữ liệu của giá trị mà hàm trả về.
   - Đây là kiểu dữ liệu của giá trị mà hàm sẽ trả về.
   - Nếu hàm không trả về gì, ta dùng từ khóa **void**.

   ```
   int getNumber();    // Hàm trả về kiểu int
   double getPrice(); // Hàm trả về kiểu double
   void showMessage(); // Hàm không trả về gì
   ```

2. **Tên hàm (Function name)**: Định danh của hàm, giúp gọi hàm dễ dàng.
   - Là định danh của hàm, dùng để gọi khi cần sử dụng.
   - Quy tắc đặt tên:
     - Không được trùng với từ khóa của C++
     - Không chứa khoảng trắng
     - Nên đặt theo quy tắc **camelCase** hoặc **snake_case**

   ```
   void printMessage(); // Tên hàm là printMessage
   int sumValues();     // Tên hàm là sumValues
   ```

3. **Danh sách tham số (Parameter list)**: Chứa các giá trị đầu vào của hàm.
   - Các giá trị đầu vào của hàm, có thể có hoặc không.
   - Tham số được đặt trong dấu ngoặc **()** và cách nhau bằng dấu **,**.
   - Có thể truyền tham số bằng giá trị, Array

   ```
   int add(int a, int b); // Hai tham số kiểu int
   void display(string name, int age); // Hai tham số: string và int
   void updateValue(int &x); // Truyền tham chiếu
   void changeArray(int p[]); // Truyền bằng array
   ```

4. **Thân hàm (Function body)**: Chứa các lệnh thực thi bên trong hàm.
   - Chứa các lệnh thực thi khi hàm được gọi.
   - Được đặt trong cặp dấu **{}**.

```
void sayHello() {
    cout << "Hello, world!";
}
```

5. **Lệnh trả về (Return statement)**: Dùng để trả về giá trị cho lời gọi hàm (nếu có).
   - Dùng từ khóa **return** để trả về giá trị cho hàm.
   - Nếu kiểu trả về là **void**, có thể không cần `return`.

```
int square(int num) {
    return num * num; // Trả về giá trị bình phương của num
}
void print(int num){
    cout << num;
    return;
}
```

3 Các loại hàm

1. **Hàm có giá trị trả về (Function with Return Value)**
   Hàm này thực hiện một công việc và trả về một giá trị.

```
int sum(int a, int b) {
    return a + b; // Trả về tổng của a và b
}

int main() {
    int result = sum(5, 7);
    cout << "Sum: " << result; // Output: Sum: 12
    return 0;
}
```

2. **Hàm không có giá trị trả về (Void Function)**
   Hàm này chỉ thực hiện công việc mà không trả về giá trị nào.

```
void greet() {
    cout << "Hello, welcome to C++!";
}

int main() {
    greet(); // Gọi hàm
    return 0;
}
```

3. **Hàm có tham số mặc định (Default Parameter Function)**
   Cho phép sử dụng giá trị mặc định nếu không có tham số truyền vào.

```
void greet(string name = "Guest") {
    cout << "Hello, " << name << "!";
}

int main() {
    greet();        // Output: Hello, Guest!
    greet("John");  // Output: Hello, John!
    return 0;
}
```

4. **Hàm đệ quy (Recursive Function)**
   Hàm này gọi lại chính nó trong quá trình thực thi.

```cpp
int factorial(int n) {
    if (n == 0) return 1; // Điều kiện dừng
    return n * factorial(n - 1);
}

int main() {
    cout << "Factorial of 5: " << factorial(5); // Output: 120
    return 0;
}
```

5. **Hàm nạp chồng (Function Overloading)**
   Cho phép nhiều hàm có cùng tên nhưng khác nhau về tham số.

```cpp
int add(int a, int b) { return a + b; }
double add(double a, double b) { return a + b; }

int main() {
    cout << add(3, 5) << endl;    // Gọi hàm add(int, int)
    cout << add(2.5, 3.5) << endl; // Gọi hàm add(double, double)
    return 0;
}
```

6. **Hàm được khai báo trước (Function Declaration/Prototype)**
   Khai báo trước giúp trình biên dịch biết về hàm trước khi sử dụng.

```cpp
// Khai báo trước (Function Prototype)
int sum(int, int);

int main() {
    int result = sum(4, 6);
    cout << "Sum: " << result; // Output: Sum: 10
    return 0;
}

// Định nghĩa hàm sau main
int sum(int a, int b) {
    return a + b;
}
```

4 Truyền tham số trong

Trong C++, tham số có thể được truyền vào hàm theo nhiều cách khác nhau. Dưới đây là bốn cách phổ biến nhất:

1. **Truyền tham trị (Pass by Value)** Khi truyền theo tham trị, hàm nhận một bản sao của giá trị được truyền vào. Việc thay đổi giá trị bên trong hàm không ảnh hưởng đến giá trị gốc.

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

void changeValue(int x) {
    x = 100;  // Thay đổi chỉ áp dụng trên bản sao của x
}

int main() {
    int a = 10;
    changeValue(a);
    cout << "Value of a: " << a << endl;  // Kết quả vẫn là 10
```

```
    return 0;
}
```

2. **Truyền tham chiếu (Pass by Reference)** Khi truyền theo tham chiếu, hàm nhận trực tiếp biến gốc, không tạo bản sao. Mọi thay đổi trong hàm sẽ tác động trực tiếp lên biến gốc.

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

void changeValue(int &x) {
    x = 100;  // Thay đổi giá trị của biến gốc
}

int main() {
    int a = 10;
    changeValue(a);
    cout << "Value of a: " << a << endl;  // Kết quả sẽ là 100
    return 0;
}
```

3. **Truyền hằng tham chiếu (Pass by Const Reference)** Dùng khi muốn truyền dữ liệu lớn vào hàm nhưng không thay đổi giá trị của nó. Giúp tránh sao chép dữ liệu mà vẫn đảm bảo an toàn.

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

void printValue(const int &x) {
    cout << "Value: " << x << endl;
}

int main() {
    int a = 10;
    printValue(a);  // Output: Value: 10
    return 0;
}
```

4. **Truyền mảng (Pass by Array)** Khi truyền một mảng vào hàm, thực chất chỉ truyền địa chỉ phần tử đầu tiên. Việc thay đổi trong hàm sẽ tác động trực tiếp lên mảng gốc.

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

void modifyArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] += 1;  // Thay đổi giá trị của mảng gốc
    }
}

int main() {
    int myArray[5] = {1, 2, 3, 4, 5};
    modifyArray(myArray, 5);

    for (int i = 0; i < 5; i++) {
        cout << myArray[i] << " ";  // Output: 2 3 4 5 6
    }
    return 0;
}
```