Functions (Hàm): Phép Thuật và Những Câu Thần Chú

Chương 5: Khám phá quyền năng của C++

Lớp học lập trình đặc biệt

Ngày 14 tháng 1 năm 2026

Lời chào mừng

Chào mừng em gia nhập lớp học lập trình đặc biệt này!

Chúng ta sẽ cùng khám phá một trong những "quyền năng"mạnh mẽ nhất của thế giới C++: **Functions (Hàm)**.

Định nghĩa theo phong cách Phép thuật

Nếu lập trình là việc em ra lệnh cho máy tính, thì **Function (Hàm)** chính là cách em dạy máy tính những "tuyệt chiêu"đặc biệt để dùng đi dùng lại mà không cần dạy lại từ đầu.

Bản đồ kho báu: Chinh phục C++ Functions

Lộ trình Khám phá

1. **Chương 1: Cuốn Sách Phép Thuật (Create a Function)**
   - Cách tạo ra một "câu thần chú"và gói ghém hành động.
2. **Chương 2: Hô Vang Câu Thần Chú (Call a Function)**
   - Cách kích hoạt sức mạnh để máy tính thực thi.
3. **Chương 3: Sức Mạnh Của Sự Tái Sử Dụng**
   - Tại sao viết code một lần nhưng dùng được cả ngàn lần?

Chương 1: Cuốn Sách Phép Thuật

Tại sao phải tạo Hàm?

Hãy tưởng tượng em chỉ huy Robot pha mì tôm. Quy trình gồm 5 bước:

1. Lấy bát.
2. Xé gói mì.
3. Đổ nước sôi.
4. Đậy nắp.
5. Chờ 3 phút.

Vấn đề: Pha cho 40 người = Viết lại 5 dòng x 40 lần = 200 dòng lệnh (Chán ngắt!).
Giải pháp: Viết 5 bước vào một cái hộp, dán nhãn `PhaMi`. Em đã tạo ra một **"Khối kiến thức"(Chunk)**.

Giải phẫu một câu thần chú

Trong C++, cú pháp để "dạy"máy tính kỹ năng mới:

```
1 void tenHam() {
2   // Nhung hanh dong cu the nam o day
3 }
```

- `void` **(Hư không):** Nghĩa là "trống rỗng". Hàm thực hiện xong là nghỉ, không cần trả lại kết quả (giá trị) cho chương trình chính.
- `tenHam` **(Tên hàm):** Nhãn dán trên hộp (Ví dụ: `PhaMi`, `XinChao`). *Lưu ý: Viết dính liền, không dấu cách.*
- `()`**:** Nơi chứa nguyên liệu đầu vào (Hiện tại đang để trống).
- `{ ... }` **(Căn Phòng Bí Mật):** Nơi chứa mã lệnh tuyệt chiêu.

Ví dụ thực tế

Tạo một câu thần chú tên là `myFunction` để in ra dòng chữ:

```
1 void myFunction() {
2   cout << "I just got executed!";
3 }
```

Ví dụ thực tế

Tạo một câu thần chú tên là `myFunction` để in ra dòng chữ:

```
1 void myFunction() {
2   cout << "I just got executed!";
3 }
```

KHOAN ĐÃ! MỘT SỰ THẬT BẤT NGỜ

Nếu em chỉ viết đoạn code trên và bấm "Chạy"(Run), **sẽ KHÔNG CÓ GÌ XẢY RA cả!**

Lý do: Em mới chỉ **VIẾT** câu thần chú vào sách. Em **CHƯA ĐỌC** nó lên! (Súng đã chế tạo nhưng chưa ai bóp cò).

Kiểm tra nhanh (Check-point)

Câu hỏi

Trong đoạn code dưới đây, đâu là "Căn phòng bí mật"chứa hành động, và đâu là "Cái nhãn tên"?

```
1 void chemHoaQua() {
2   cout << "Chem dua hau!";
3   cout << "Chem dua!";
4 }
```

Kiểm tra nhanh (Check-point)

Câu hỏi

Trong đoạn code dưới đây, đâu là "Căn phòng bí mật"chứa hành động, và đâu là "Cái nhãn tên"?

```
1 void chemHoaQua() {
2   cout << "Chem dua hau!";
3   cout << "Chem dua!";
4 }
```

Đáp án

- **Tên hàm (Nhãn):** `chemHoaQua`
- **Căn phòng bí mật:** Cặp ngoặc nhọn `{ }` và nội dung bên trong.

Cặp `()` giống như công tắc để bật cái tên đó lên.

Chương 2: Hô Vang Câu Thần Chú

Sân Khấu Chính: `main()`

- **Hàm `main()`**: Sân Khấu Trung Tâm. Chương trình luôn bắt đầu từ đây.
- **Hàm của em (`myFunction`)**: Ca sĩ đứng trong cánh gà.

Để kích hoạt, em phải viết tên hàm kèm dấu `();` bên trong `main()`. Đây là **Calling a function**.

```
1  // 1. CANH GA (Dinh nghia)
2  void xinChao() {
3    cout << "Xin chao cac ban!";
4  }
5
6  // 2. SAN KHAU CHINH (Chay)
7  int main() {
8    xinChao(); // <--- HO THAN CHU!
9    return 0;
10 }
```

Sức mạnh của việc gọi nhiều lần

Trong `main()`, em có thể hô câu thần chú bao nhiêu lần tùy thích (giống như nút Replay).

```
int main() {
  xinChao();
  xinChao();
  xinChao();
  return 0;
}
```

Thay vì viết lại lệnh cout 3 lần, em chỉ cần gọi tên hàm 3 lần.

Thử thách tư duy (Mind Challenge)

Tình huống

Màn hình sẽ in ra chính xác những gì? Hãy đóng vai robot và chạy từng dòng lệnh.

```
1  void voTay() {
2    cout << "Bop! ";
3  }
4
5  int main() {
6    cout << "Bat dau: ";
7    voTay();
8    voTay();
9    cout << "Het bai.";
10   return 0;
11 }
```

Thử thách tư duy (Mind Challenge)

Tình huống

Màn hình sẽ in ra chính xác những gì? Hãy đóng vai robot và chạy từng dòng lệnh.

```
1  void voTay() {
2    cout << "Bop! ";
3  }
4
5  int main() {
6    cout << "Bat dau: ";
7    voTay();
8    voTay();
9    cout << "Het bai.";
10   return 0;
11 }
```

Kết quả

"Bắt đầu: Bốp! Bốp! Hết bài."

1. In "Bắt đầu: "

Chương 3: Sức Mạnh Của Sự Tái Sử Dụng

> *"Tôi chọn người lười để làm việc khó, vì họ sẽ tìm ra cách dễ nhất để làm nó. Bill Gates*

Hàm là công cụ giúp lập trình viên "lười một cách vĩ đại".

Năng lực 1: Bảo Trì (Sửa 1 được 100)

Ví dụ: Chuỗi 100 cửa hàng trà sữa. Đổi công thức từ "trân châu đen"sang "trân châu trắng".

- **Không dùng hàm:** Phải sửa code ở 100 nơi (Copy/Paste).
- **Dùng hàm:** Chỉ sửa trong hàm `PhaTraSua` tại trụ sở chính. 100 cửa hàng tự động cập nhật.

Năng lực 2: Sự Gọn Gàng (Chia để trị)

Thay vì viết 1000 dòng code lộn xộn trong `main()`, chúng ta chia nhỏ vấn đề:

```
int main() {
    xayMongNha();    // Ham xay mong
    dungCotNha();    // Ham dung cot
    lopMaiNha();     // Ham lop mai
    sonTuong();      // Ham son tuong
    return 0;
}
```

Nhìn vào đây, ai cũng hiểu quy trình xây nhà mà không cần biết chi tiết trát vôi vữa thế nào (Tư duy trừu tượng hóa).

Tổng kết hành trình

Chúc mừng em đã hoàn thành khóa học cấp tốc về Functions!

1. **Tạo Hàm (`void tenHam() {...}`):** Gói ghém hành động vào hộp bí mật.
2. **Gọi Hàm (`tenHam();`):** Hô thần chú để kích hoạt từ `main()`.
3. **Tại sao dùng Hàm:** Tái sử dụng code, dễ sửa lỗi (Sửa 1 được 100), code gọn gàng.

Bài tập cuối khóa (Final Boss)

Tình huống Game Bắn Súng

- Em có hàm `banSung()` chứa 5 dòng lệnh tính toán đạn bay.
- Nhân vật bắn 500 lần trong game.
- Đột nhiên phát hiện công thức sai.

Câu hỏi:

1. Nếu KHÔNG dùng hàm (Copy-Paste): Phải sửa bao nhiêu dòng code?
2. Nếu DÙNG hàm: Phải sửa bao nhiêu chỗ?

Bài tập cuối khóa (Final Boss)

Tình huống Game Bắn Súng

- Em có hàm `banSung()` chứa 5 dòng lệnh tính toán đạn bay.
- Nhân vật bắn 500 lần trong game.
- Đột nhiên phát hiện công thức sai.

Câu hỏi:

1. Nếu KHÔNG dùng hàm (Copy-Paste): Phải sửa bao nhiêu dòng code?
2. Nếu DÙNG hàm: Phải sửa bao nhiêu chỗ?

Đáp án

1. **Không dùng hàm:** Phải sửa 500 chỗ khác nhau (tương đương tìm và sửa 2500 dòng lệnh). Rất dễ sót lỗi!
2. **Dùng hàm:** Chỉ cần sửa đúng **1 chỗ** duy nhất (trong định nghĩa hàm `banSung`).