KỸ THUẬT LẬP TRÌNH C/C++

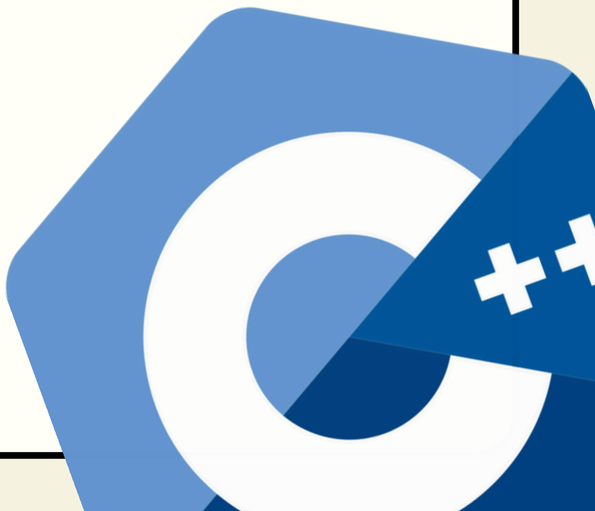
Dạy bởi 1 SVBK

Đại học Bách Khoa Hà Nội
Kỹ thuật Điện tử - Viễn thông

2024-25

NỘI DUNG

- Hàm
- Tham số & Đối số



GIỚI THIỆU CHUNG

Một **hàm** trong C là một **tập hợp các câu lệnh thực hiện một nhiệm vụ cụ thể khi được gọi**. Đây là khối xây dựng cơ bản của một chương trình C, giúp tăng tính **module hóa (modularity)** và **tái sử dụng mã nguồn (code reusability)**. Các câu lệnh trong một hàm được bao bọc trong **dấu ngoặc nhọn { }**, mỗi câu lệnh có ý nghĩa và thực hiện các thao tác nhất định. Trong các ngôn ngữ lập trình khác, hàm cũng có thể được gọi là **tiểu trình (subroutine)** hoặc **thủ tục (procedure).**

Trong chương này, chúng ta sẽ tìm hiểu về hàm, cách **định nghĩa (definition)** và **khai báo (declaration)** hàm, các **tham số (parameters)** và **đối số (arguments)**, **giá trị trả về (return values)**, và nhiều khía cạnh khác liên quan đến hàm trong C.

GIỚI THIỆU CHUNG

LỢI ÍCH CỦA HÀM

- CODE DỄ ĐỌC
- DỄ DEBUG
- DỄ THAY ĐỔI
- TÁI SỬ DỤNG

HÀM (FUNCTION)

HÀM ĐỊNH NGHĨA SẴN

Trong suốt quá trình học từ đầu đến giờ, ta đã luôn biết về hàm, chúng ta đã sử dụng nhiều hàm mà không hề nhận ra.

Trong ví dụ sau, **hàm main()** là để thực thi mã lệnh, **hàm printf()** là hàm được sử dụng để in dòng chữ ra màn hình:

```c
#include <stdio.h>

int main () {
    printf("Hello World");
}
```

HÀM (FUNCTION)

HÀM ĐỊNH NGHĨA SẴN

Trong C, **hàm được định nghĩa sẵn (predefined functions)** là những hàm có sẵn trong thư viện chuẩn của ngôn ngữ, giúp lập trình viên thực hiện các tác vụ phổ biến mà không cần tự viết mã từ đầu. Những hàm này thuộc các thư viện như stdio.h, math.h, string.h, v.v.

Dưới đây là một số ví dụ về các nhóm hàm được định nghĩa sẵn trong C:

- **printf()** – Xuất dữ liệu ra màn hình
- **scanf()** – Nhập dữ liệu từ bàn phím
- **gets()** – Nhập một chuỗi ký tự
- **puts()** – Xuất một chuỗi ký tự
- Và các **hàm toán học** trong **thư viện math.h** ta đã được được học...

HÀM (FUNCTION)

XÂY DỰNG HÀM

Cú pháp xây dựng hàm:

```
kiểu_trả_về tên_hàm(danh sách tham số) {
    // các biểu thức
}
```

Ví dụ về xây dựng hàm:

```
int dienTich(int x, int y) {
    return x * y;
}

int main() {
    int x = 5;
    int y = 10;
    printf("%d", dienTich(x, y));
    return 0;
}
```

HEADER

BODY
(THÂN HÀM)

HÀM (FUNCTION)

XÂY DỰNG HÀM

CẤU TẠO CỦA HÀM

Hàm gồm 2 thành phần chính là **header & body.** Trong header có:

- **Kiểu trả về:** có thể là **kiểu dữ liệu nguyên thủy** như int, double, char,... hoặc nếu **không trả về giá trị** → ta sử dụng **kiểu dữ liệu void.**
- **Tên hàm:** tên của hàm, cần tuân thủ nguyên tắc đặt tên.
- **Danh sách tham số (parameter list):** là một danh sách các tham số được **phân tách bằng dấu phẩy**, xác định các tham số mà hàm nhận khi được gọi. Nếu một hàm không nhận bất kỳ tham số nào, danh sách tham số nên chứa từ khóa void. Mỗi tham số phải bao gồm kiểu dữ liệu của nó; nếu không, lỗi biên dịch sẽ xảy ra.

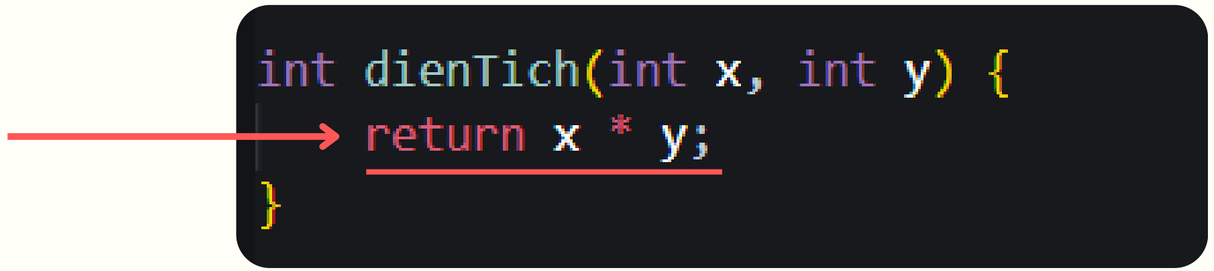
HÀM (FUNCTION)

XÂY DỰNG HÀM

CẤU TẠO CỦA HÀM

Tiếp theo là thân hàm, các câu lệnh **bên trong dấu ngoặc nhọn** tạo thành thân hàm **(function body)**, cũng chính là một **khối lệnh (block)**. Các **biến cục bộ (local variables)** có thể được khai báo trong bất kỳ khối nào, và các khối có thể được lồng nhau. Tuy nhiên, các hàm không thể được lồng nhau — việc định nghĩa một hàm bên trong một hàm khác là một lỗi cú pháp.

thân hàm, gồm câu lệnh trả về kết quả của phép nhân



HÀM (FUNCTION)

KIỂU TRẢ VỀ CỦA HÀM

Trong C, **kiểu trả về của hàm (return type)** là kiểu dữ liệu mà **hàm sẽ trả về** sau khi thực hiện xong công việc của nó. Việc chọn kiểu trả về phụ thuộc vào mục đích của hàm và cách ta muốn sử dụng kết quả. Một số kiểu trả về ta có thể thường gặp:

- **void (không trả về gì):** hàm sẽ chỉ thực hiện **một công việc** (ví dụ: in ra màn hình, thay đổi giá trị biến toàn cục,...) mà không cần gửi kết quả về nơi gọi hàm.
- Trả về **kiểu dữ liệu nguyên thủy** (int, double, char,...): hàm sẽ thực hiện **tính toán hoặc xử lý dữ liệu** và **trả kết quả** về nơi gọi hàm. Kết thúc hàm ta phải dùng từ khóa **return** để trả về một giá trị **đúng kiểu đã khai báo.** Nếu không, trình biên dịch sẽ **báo lỗi hoặc gây hành vi không xác định.**
- Và 1 số kiểu trả về khác như struct, union, con trỏ,...

HÀM (FUNCTION)

KIỂU TRẢ VỀ CỦA HÀM

VẬY KHI NÀO DÙNG CÁI NÀO?

- **void:** Dùng khi hàm không cần trả về giá trị nào**, chỉ thực hiện tác vụ** (ví dụ: in ra màn hình, ghi file, v.v.).
- **Kiểu số (int, float, double):** Dùng khi hàm **thực hiện phép tính và trả về kết quả** số học. int thường dùng cho kết quả nguyên, float hoặc double dùng cho kết quả có phần thập phân.
- **char:** Dùng khi hàm trả về **một ký tự duy nhất.**

HÀM (FUNCTION)

GỌI HÀM (CALL FUNCTION)

Sau khi xây dựng hàm, việc tiếp theo chúng ta cần là **thực thi hàm**, để làm được vậy ta cần **vào hàm main()** và **gọi hàm chúng ta vừa xây dựng xong khi cần.** Các câu lệnh trong hàm sẽ được thực hiện, và sau khi hàm kết thúc, chương trình sẽ **tiếp tục thực thi các câu lệnh bên dưới hàm.**

Ta sẽ xem xét ví dụ sau, đi từ **xây dựng hàm** cho đến **gọi và thực thi hàm** với công việc **tính diện tích hình chữ nhật,** hàm có **2 tham số** tương ứng với **chiều dài và chiều rộng:**

HÀM (FUNCTION)

```c
int dienTich(int x, int y) {
    return x * y;
}

int main () {
    int x = 5;
    int y = 10;
    printf("%d", dienTich(x, y));
    return 0;
}
```

```
50
```

GỌI HÀM (CALL FUNCTION)

Khai báo hàm **dienTich()** với **kiểu trả về int**, các tham số là **x** (int) và **y** (int). Hàm sẽ trả về kết quả là **giá trị** phép nhân giữa x với y.

Truyền **đối số** khi gọi hàm. Đối số là các **giá trị thực tế** ta "đưa" vào tham số và **gán lần lượt** cho các tham số tương ứng.

Ta sẽ in giá trị được trả về ra **qua hàm printf().**

HÀM (FUNCTION)

GỌI HÀM (CALL FUNCTION)

```c
void ten(char s[]) {
    printf("%s", s);
}

int main () {
    char s[50] = "Colleen";
    ten(s);
    return 0;
}
```

```
Colleen
```

Khai báo hàm **ten()** với tham số là 1 chuỗi ký tự s[], hàm không có kiểu trả về và, khi được gọi

Khai báo chuỗi s ban đầu.

Ta sẽ in giá trị được trả về ra **qua hàm printf().**