**Tìm hiểu về lập trình C cơ bản.**

1. Ngôn ngữ C là gì, ưu và nhược điểm, phạm vi ứng dụng?

* Là ngôn ngữ lập trình hệ thống
* Là ngôn ngữ thủ tục
* Có cấu trúc rõ ràng: C có cú pháp đơn giản và cấu trúc dễ đọc, với các khối mã được đặt trong cặp dâu ngoặc nhọn: {}
* Ngôn ngữ lập trình cấp trung, hỗ trợ tính năng cho cả ngôn ngữ cấp thấp và ngôn ngữ cấp cao
* Cơ sở cho các ngôn ngữ lập trình khác => ngôn ngữ mẹ
* Sử dụng để lập trình cấp thấp, tạo ra các thiết bị phần cứng, hệ điều hành, trình điều khiển, nhân hệ điều hành
* Ưu điểm:

+ Linh hoạt: C cho phép truy cập trực tiếp vào bộ nhớ máy tính, điều này làm cho nó rất mạnh mẽ và linh hoạt. Tuy nhiên, cũng có thể dẫn đến các lỗi như tràn bộ nhớ đệm hoặc con trỏ không hợp lệ

+ Nhiều kiểu dữ liệu và toán tử có sức ảnh hưởng

+ Có tính khả chuyển cao, dễ dàng chạy trên máy tính khác mà không cần thực hiện bất cứ thay đổi nào hoặc cần sửa đổi nhưng rất đơn giản.

+ Khả năng tự mở rộng, dễ dàng thêm các hàm của người dùng

+ Ngôn ngữ lập trình có cấu trúc, hệ thống

+ Hiệu suất cao, có tốc độ nhanh vì không có các thủ tục bổ sung

* Nhược điểm

+ Không có lập trình hướng đối tượng

+ Không kiểm tra thời gian chạy

+ Không kiểm tra loại nghiêm ngặt, không an toàn về kiểu dữ liệu

+ Không có khái niệm không gian tên

+ Không có khái niệm hàm tạo/hủy

+ Không hỗ trợ tự động quản lý bộ nhớ

* Phạm vi ứng dụng

+ Lập trình hệ thống

+ Lập trình nhúng

+ Lập trình ứng dụng

1. Biến, hằng, toán tử, macro, kiểu dữ liệu

* Biến (variables):

+ Vùng bộ nhớ được cấp phát để lưu trữ dữ liệu

+ Mỗi biến có một kiểu các định, kích thước

* Hằng (constant)

+ Các giá trị không thay đổi trong suốt quá trình chạy chương trình

+ Khai báo bằng từ khóa const

* Toán tử (operation)

+ Ký hiệu hoặc ký tự đặc biệt

+ Được sử dụng để thực hiện các phép tính hoặc thao tác trên biến và giá trị

+ Toán tử số học, gán, so sánh, logic, tăng\giảm, sizeof, condition

* Macro:

+ Cho phép người dùng định nghĩa hằng số hoặc chuỗi mã thay thế

+ Tối ưu hóa mã: thay thế các giá trị cố định trong mã nguồn

+ Dễ dàng duy trì: thay đổi giá trị của macro chỉ cần thay đổi một lần

* Kiểu dữ liệu (variable type):

+ Vai trò xác định và quản lý các loại dữ liệu khác nhau trong chương trình

+ Xác định các giá trị mà một biến có thể nhận. giá trị mà một hàm có thể trả về

Có thể tự định nghĩa kiểu dữ liệu sử dụng typedef

1. Các cấu trúc điều kiện, vòng lặp.

* Cấu trúc điều kiện

+ Cho phép thay đổi luồng chương trình dựa trên điều kiện nào đó

+ if, if-else, switch

* Cấu trúc vòng lặp

+ Thực hiện một đoạn mã hoặc nhóm lệnh nhiều lần

while, do-while, for

+ Có thể lồng các vòng lặp trong nhau để thực hiện chương trình phức tạp

1. Cấu trúc dữ liệu tự định nghĩa: Struct, union

* Struct:

+ Cho phép lưu trữ các loại phần tử khác nhau

+ Có thể truy cập tất cả các thành phần của struct trong cùng một thời điểm của chương trình

* Union:

+ Cho phép lưu trũ các kiểu dữ liệu khác nhau trong một vùng nhớ

+ Tiết kiệm bộ nhớ bằng các sử dụng cùng một vùng nhớ cho nhiều mục đích

+ Chỉ có thể truy cập một thành phần của struct trong cùng một thời điểm của chương trình

1. Con trỏ, mảng.

* Con trỏ

+ Cho phép người dùng thao tác trực tiếp với bộ nhớ

+ Làm việc với địa chỉ của các biến

+ Không lưu giá trị bình thường mà trỏ tới địa chỉ

* Mảng

+ Cho phép lưu trữ tập hợp các phần tử cùng dữ liệu

+ Mảng 1 chiều, mảng đa chiều

1. Hàm, tham chiếu, tham trị

* Hàm

+ Cho phép tổ chức mã nguồn một cách hiệu quả, cấu trúc rõ ràng, dễ quản lý

+ Cho phép tái sử dụng

+ Thực hiện một nhiệm vụ cụ thể

+ Nhận tham số đầu vào (nếu có) và thực hiện một chức năng cụ thể, kết quả (nếu có) được trả về cho hàm gọi

* Tham chiếu (pass by reference)

+ Truyền địa chỉ ô nhớ của biến vào hàm

+ Hàm làm việc trực tiếp với biến gốc thông qua địa chỉ

+ Thay đổi giá trị của biến trong hàm cũng khiến giá trị của biến gốc thay đổi

* Tham trị (pass by value)

+ Truyền giá trị của biến vào hàm

+ Hàm tạo ra bản sao của giá trị và làm việc với bản sao

+ Thay đổi giá trị của biến trong hàm không khiến giá trị của biến gốc thay đổi

**Embedded C**

1. Sự khác nhau giữa C thông thường với Embedded C

* Embedded C:

+ Giao tiếp trực tiếp với bộ xử lý đích, cho phép người dùng tương tác trực tiếp với phần cứng

+ Hiệu suất tối ưu cho ứng dụng nhúng

+ Không phải là một phần của ngôn ngữ C thông thường

+ Không phù hợp để xử lý máy tính để bàn

1. Lưu ý khi lập trình C embedded

* Hạn chế về bộ nhớ: tài nguyên ROM và Ram thường hạn chế
* Phụ thuộc phần cứng
* Đáp ứng chính xác về thời gian: việc xử lý ngắt (interrupt) và các tác vụ khác thường đòi hỏi đáp ứng chính xác về thời gian
* Kiểu pointer đa dạng
* Các từ khóa và token đặc biệt

1. C memory management

* Cấp phát động (allocating memory dynamically)

+ Giải quyết vấn đề thiếu bộ nhớ bằng cách xin cấp phát thêm bộ nhớ trong thời gian chạy chương trình

+ Tiết kiệm bộ nhớ

+ Linh hoạt, cho phép quản lý, phân bổ và giải phóng bộ nhớm trong khi chạy chương trình

+ malloc(), calloc(), free(), realloc()

1. Compilation process, toolchain, compiler, linker.

* Compilation process: biên dịch -> liên kết -> gán địa chỉ vật lý
* Toolchain: trình biên dịch, trình liên kết, trình tạo thư viện, trình kiểm tra mã nguồn, trình chuyển mã
* Compiler:

+ Giúp chuyển đổi mã nguồn từ ngôn ngữ lập trình thành mã máy mà hệ thống nhúng có thể hiểu và thực thi

+ Tạo mã máy tối ưu, thời gian thực thi nhanh

+ Cho phép người dùng tương tác với phần cứng

+ Đảm bảo tính đúng đắn

* Linker

+ Công cụ trong quá trình biên dịch thành mã máy thực thi

+ Kết hợp các file đối tượng đã được biên dịch từ mã nguồn thành file hoàn chỉnh

+ Giải quyết các tham chiếu giữa hàm và biến trong các file đối tượng khác nhau

+ Thực hiện việc đặt hàm và biến vào vị trí chính xác

+ Static linking, dynamic linking

+ Quản lý việc liên kết với thư viện

+ Tách biệt mã nguồn và mã máy, giảm kích thước file

+ Cho phép tái sử dung

+ Đảm bảo tính đúng đắn và hiệu suất chương trình

1. Make, build system: lập trình hệ thống nhúng

* Xây dựng các hệ thống phần mềm để thực hiện kết nối và điều khiển các thành phần trong hệ thống nhúng
* Thực hiện các chức năng chuyên trách hoặc riêng biệt nào đó
* Ứng dụng để giải quyết các bài toán chuyên dụng
* Ràng buộc về thời gian thực
* Tối ưu hóa sản xuất

1. 8bit/16bit/32bit machine

* Độ rộng dữ liệu
* Càng lớn càng tiêu tốn nhiều tài nguyên

1. Tổ chức firmware, kiến trúc phần mềm.

* Firmware:

+ Một dạng của embedded software

+ Được lưu trữ trên chip lưu trữ

+ Sử dụng để điều khiển các phần cứng cố định trên thiết bị

+ Được cài đặt 1 lần và giữ nguyên trong suốt tuổi thọ của thiết bị

* Kiến trúc phần mềm

+ Tổ chức cơ bản mà một hệ thống thể hiện trong các thành phần của nó, mối quan hệ của chúng với nhau và với môi trường

+ Là bản thiết kế để từ đó triển khai phần mềm

+ Cung cấp cho nhà phát triển từng thành phần và cấu trúc phần mềm chính

+ Vai trò quan trọng trong tổ chức và quản lý phần mềm