Tìm hiểu về lập trình C cơ bản.

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

-Ngôn ngữ lập trình C là một trong những ngôn ngữ cơ bản, sơ cấp cho người mới bắt đầu làm quen với lập trình. Loại ngôn ngữ này cung cấp cho người học quyền truy cập cấp thấp vào bộ nhớ của hệ thống.

-Ưu điểm:

+Khối xây dựng cho nhiều ngôn ngữ lập trình khác

+Ngôn ngữ mạnh mẽ và hiệu quả

+Ngôn ngữ di động

+Chức năng tích hợp sẵn

+Chất lượng để tự mở rộng

+Ngôn ngữ lập trình có cấu trúc, ở cấp độ trung bình

+Thực hiện các thuật toán và cấu trúc dữ liệu

+Ngôn ngữ lập trình thủ tục

+Cấp phát bộ nhớ động

+Lập trình hệ thống

-Nhược điểm

+Không hỗ trợ OOP

+Kiểm tra thời gian chạy

+Khái niệm về không gian tên

+Thiếu xử lý ngoại tệ

+Hàm tạo hoặc hủy

+Mức độ trừu tượng thấp

-Phạm vi ứng dụng:

+Hệ thống nhúng: được sử dụng rộng rãi trong lập trình nhúng vì khả năng tiếp cận cấp thấp và hiệu suất cao

+Hệ điều hành: nhiều hệ điều hành như Unix, Linux được phát triển bằng C và các ứng dụng hệ thống và phần mềm cần hiệu suất cao như trình biên dịch, trình gỡ lối,..

+Phát triển phần mềm hệ thống: thường được sử dụng để phát triển các ứng dụng hệ thống và phần mềm cần hiệu suất cao

+Trò chơi và ứng dụng đồ họa: mặc dù không phải là lựa chọn phổ biến nhưng vẫn có thể sử dụng C để phát triển các trò chơi và ứng dụng đồ họa

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

-Biến là nơi lưu trữ tạm thời dữ liệu với 1 kiểu dữ liệu xác định. Biến có thể thay đổi giải trụ trong quá trình thực thi chương trình

Trong ngôn ngữ C, biến có 3 thành phần:

+Kiểu dữ liệu:

+Tên gọi

+Giá trị

Trong ngôn ngữ lập trình C, mọi biến khi khai báo đều có 2 thuộc tính: kiểu dữ liệu (type) và lớp lưu trữ (storage class) của nó. Lớp lưu trữ ở đây chính là thuộc tính thể hiện phạm vi của biến - nơi nào có thể dùng biến đó và biến đó tồn tại trong bao lâu. Có 2 loại phạm vi chính cần quan tâm như sau:

+Biến toàn cục

+Biến cục bộ

-Hằng: là biến có giá trị không đổi (khởi tạo bằng 2 cách)

+ sử dụng bộ tiền xử lý: #define

+ sử dụng từ khóa: const

- Marco là 1 tên bất kì (do lập trình viên đặt tên) trỏ tới 1 khối lệnh thực hiện một chức năng nào đó.

-Trong quá trình tiền xử lí (pre-processor), các macro được sử dụng trong chương trình được thay thế bởi các khối câu lệnh tương ứng.

-Định nghĩa macro bằng lệnh #define

Kiểu dữ liệu

-Kiểu dữ liệu nguyên thủy (primitive data types): int, char, float,…

-Kiểu dữ liệu phức tạp (complex data types): array, struct, union, enum,..

Các kiểu dữ liệu này cung cấp sự linh hoạt và tiện ích cho việc lập trình trong ngôn ngữ C, cho phép bạn xử lý nhiều loại dữ liệu khác nhau một cách hiệu quả.

1. Các cấu trúc điều kiện, vòng lặp.

-If statement

Câu lệnh điều kiện "if" trong ngôn ngữ lập trình C cho phép bạn kiểm tra một điều kiện nào đó và thực hiện các hành động khác nhau dựa trên kết quả của điều kiện đó.

If(condition)

{

Statement

}

-If-else statement: condition đúng thực hiện khối lệnh bên trong còn sai thì chuyển sang khối lệnh else

If(condition)

{

Statement

}

else

{

statement

}

-If-else if-else statement: sử dụng khi cần kiểm tra nhiều điều kiện trong chương trình

If (condition1){

Statement //condtion1 dung

} else if(condition2){

Statement //condtion 2 dung

}…

Else{

Statement// khi tất cả điều kiện trên đều sai

}

-Loop

+while

+for

+do while

+Lồng các vòng lặp

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

-Struct: được sử dụng để định nghĩa kiểu dữ liệu mới dựa trên kiểu dữ liệu hiện có và có đặc điểm tập hợp được nhiều thành phần có kiểu dữ liệu khác nhau. Bộ nhớ cho struct sẽ là tổng bộ nhớ cần thiết cho các phần tử bên trong nó.

-Union: là 1 kiểu dữ liệu được định nghĩa bởi người dùng cho phép lưu trữ dữ liệu của nhiều khác nhau trong cùng 1 vùng nhớ. Mỗi phần tử của union sẽ chia sẻ cùng 1 địa chỉ bắt đầu, điều này có nghĩa là union sẽ chiếm diện tích bộ nhớ bằng phần tử lớn nhất của nó.

1. Con trỏ, mảng.

-Con trỏ: là một biến đặc biệt trong ngôn ngữ lập trình C, nó lưu trữ địa chỉ bộ nhớ của một biến khác. Sử dụng con trỏ, bạn có thể thực hiện các thao tác như truy cập, thay đổi giá trị của biến, cũng như quản lý vùng nhớ động.

Việc sử dụng con trỏ trong ngôn ngữ lập trình C mang lại nhiều lợi ích, bao gồm: quản lý bộ nhớ động, truyền tham chiếu, đa chiều và mảng động, hiệu suất, cấu trúc dữ liệu phức tạp, điều khiển dữ liệu nhanh chóng.

-Mảng: là một tập hợp các phần tử có cùng kiểu dữ liệu được sắp xếp theo thứ tự liên tục trong bộ nhớ. Mảng cho phép lưu trữ nhiều giá trị dữ liệu trong cùng một biến và truy cập đến chúng thông qua chỉ số.

Trong ngôn ngữ lập trình C, mảng và con trỏ có một mối quan hệ chặt chẽ với nhau. Thực tế, mỗi biến mảng trong C thực chất là một con trỏ đến phần tử đầu tiên của mảng đó.

1. Hàm, tham chiếu, tham trị

-Hàm: là một khối mã được đặt tên và có thể tái sử dụng để thực hiện một tác vụ cụ thể. Việc sử dụng hàm giúp tăng cấu trúc và sắp xếp mã của chương trình, làm cho mã dễ đọc hơn và giúp tái sử dụng mã một cách hiệu quả.

-Tham chiếu và tham trị là 2 cách để truyền tham số cho hàm. Trong đó:

+Trong tham trị, giá trị của biến được sao chép và truyền vào hàm

+Hàm chỉ có quyền truy cập và thay đổi bản sao của giá trị này, không ảnh hưởng đến giá trị của biến gốc ben ngoài hàm

+Không thể thay đổi giá trị của biến gốc

+Trong tham chiếu, địa chỉ của biến được truyền vào hàm

+Hàm có thể truy cập và thay đổi giá trị của biến gốc thông qua địa chỉ này

+Có thể thay đổi giá trị của biến gốc

1. Embedded C
2. Sự giống và khác nhau giữa C thông thường với Embedded C

-Giống nhau:  
+Cú pháp cơ bản: đều dùng cú pháp cơ bản của lập trình C

+Kiểu dữ liệu và cấu trúc điều khiển

+Thư viện tiêu chuẩn

-Khác nhau:

+Môi trường mục tiêu: C thông thường thường được sử dụng cho các ứng dụng phần mềm trên các hệ điều hành thông thường như Windows, Linux, hoặc macOS. Trong khi đó, Embedded C được sử dụng cho việc phát triển phần mềm nhúng, trong đó các ứng dụng chạy trên các hệ thống nhúng như vi điều khiển, vi xử lý, hoặc hệ thống nhúng khác.

+Hạn chế phần cứng: Trong lập trình nhúng, có hạn chế về tài nguyên phần cứng như bộ nhớ và tốc độ xử lý. Embedded C thường phải xử lý các hạn chế này và tối ưu hóa mã để hoạt động hiệu quả trên các thiết bị nhúng

+Thư viện hỗ trợ: Embedded C thường sử dụng các thư viện và API được tối ưu hóa cho việc lập trình nhúng, như thư viện điều khiển phần cứng (hardware abstraction layers - HALs) và thư viện hỗ trợ việc giao tiếp như UART, SPI, I2C.

+Quy trình phát triển và kiểm tra: Trong lập trình nhúng, quy trình phát triển và kiểm tra thường phức tạp hơn do việc phải xử lý vấn đề về phần cứng, nơi bạn cần phải kiểm tra chức năng của mã trên môi trường thực tế.

1. Lưu ý khi lập trình C embedded

-Hiểu rõ phần cứng: Để lập trình hiệu quả cho thiết bị nhúng, bạn cần hiểu rõ về phần cứng mà bạn đang làm việc, bao gồm vi điều khiển, vi xử lý, cấu trúc bộ nhớ và các giao diện ngoại vi như UART, SPI, I2C.

-Tối ưu hóa tài nguyên: Trong lập trình nhúng, tài nguyên như bộ nhớ và tài nguyên tính toán thường có hạn chế. Hãy tối ưu hóa mã của bạn để sử dụng tài nguyên này một cách hiệu quả.

-Điều khiển cấu trúc và cấu hình: Đảm bảo rằng bạn hiểu rõ về điều khiển cấu trúc và cấu hình của thiết bị nhúng. Điều này bao gồm việc thiết lập thanh ghi và các thiết lập khác để cấu hình các chức năng phần cứng như ngắt, giao diện, và bộ định thời.

-Kiểm tra gỡ lối: Quy trình kiểm tra và gỡ lỗi trong lập trình nhúng thường khác biệt so với lập trình phần mềm thông thường. Sử dụng công cụ như debuggers, logic analyzers, và emulators để kiểm tra và gỡ lỗi mã của bạn trên môi trường thực tế.

-Xử lý tương thích thời gian thực: Trong một số trường hợp, việc phải xử lý các tác vụ trong thời gian thực có thể là một yêu cầu quan trọng. Hãy cân nhắc các phương pháp xử lý tương thích với thời gian thực như việc sử dụng ngắt hoặc các giải pháp đồng bộ khác.

-Bảo mật và an toàn: Trong một số ứng dụng nhúng, bảo mật và an toàn có thể là yếu tố quan trọng. Hãy cân nhắc các biện pháp bảo mật như mã hóa dữ liệu, kiểm tra và xác thực đầu vào, và cách xử lý lỗi an toàn.

-Tài liệu cộng đồng: Tìm hiểu và sử dụng các tài liệu hỗ trợ từ nhà sản xuất vi điều khiển hoặc từ cộng đồng lập trình nhúng. Cộng đồng lập trình nhúng rất lớn và có thể cung cấp nhiều hỗ trợ và giải pháp cho các vấn đề phức tạp.

1. C memory management

Bộ nhớ tĩnh và động

-Bộ nhớ tĩnh: Biến được khai báo với từ khóa static hoặc nằm ngoài các hàm (global scope) sẽ được cấp phát trong vùng bộ nhớ tĩnh và tồn tại suốt thời gian chạy của chương trình.

-Bộ nhớ động: Bộ nhớ được cấp phát và giải phóng tại thời điểm chạy của chương trình sử dụng các hàm như malloc(), calloc(), realloc() và free(). Bộ nhớ động cho phép bạn cấp phát bộ nhớ tùy thuộc vào nhu cầu của chương trình.

Hàm quản lý bộ nhớ trong C  
-malloc(): Dùng để cấp phát một khối bộ nhớ có kích thước nhất định.

-calloc() : Tương tự như malloc() nhưng nó cấp phát và khởi tạo tất cả các byte được cấp phát thành giá trị 0.

-free(): Dùng để giải phóng bộ nhớ đã được cấp phát trước đó.

-realloc(): Dùng để thay đổi kích thước của khối bộ nhớ đã được cấp phát trước đó.

1. Compilation process, toolchain, compiler, linker.

- Quá trình biên dịch (compilation process) là quá trình chuyển đổi mã nguồn từ ngôn ngữ lập trình thành mã máy hoặc mã thực thi mà máy tính có thể hiểu và thực thi. Quá trình này thường bao gồm các bước như: phân tích cú pháp, sử dụng trình dịch, biên dịch, liên kết

-Toolchain: Một toolchain là một tập hợp các công cụ cần thiết để biên dịch và xây dựng một chương trình từ mã nguồn. Các thành phần chính của một toolchain bao gồm trình biên dịch (compiler), trình liên kết (linker), và các công cụ khác như trình dịch (preprocessor) và bộ tạo (builder).

-Trình biên dịch (compiler) là một chương trình máy tính dùng để chuyển đổi mã nguồn từ một ngôn ngữ lập trình thành mã máy hoặc mã trung gian. Compiler phân tích cú pháp của mã nguồn và tạo ra mã nhị phân tương ứng.

-Trình liên kết (linker) là một chương trình máy tính dùng để kết hợp các tệp nhị phân đã được biên dịch thành một chương trình hoàn chỉnh. Linker thực hiện việc giải quyết các tham chiếu giữa các tệp nhị phân và thư viện, và tạo ra một tệp nhị phân thực thi có thể chạy được.

1. Make, build system.

‘Make’ là một công cụ phổ biến trong quản lý và tự động hóa quy trình biên dịch và xây dựng (build process) của một dự án phần mềm. Một hệ thống xây dựng (build system) thường sử dụng ‘Makefile’ để mô tả các quy tắc và phụ thuộc giữa các tệp trong dự án. Dưới đây là một số khái niệm cơ bản về ‘Make’ và hệ thống xây dựng:

1. ‘Makefile’: là một tệp văn bản đặc biệt chứa các quy tắc và hướng dẫn cho công cụ ‘make’. Trong ‘Makefile’, bạn có thể định nghĩa các tên mục tiêu (targets) và quy tắc (rules) cho việc biên dịch các tệp mã nguồn và xây dựng chương trình.
2. Target: là một mục đích cụ thể mà bạn muốn make thực hiện. Mỗi mục tiêu có thể là một tên tệp, tên chương trình hoặc một hành động cụ thể như "clean" (xóa các tệp đã tạo ra trong quá trình xây dựng).
3. Rule: xác định cách tạo ra một mục tiêu. Nó bao gồm một tên mục tiêu, các phụ thuộc (dependencies) và các hành động (actions) cần thực hiện để tạo ra mục tiêu đó.
4. Dependency: là các tệp hoặc các mục tiêu mà một mục tiêu cụ thể phụ thuộc vào. Nếu bất kỳ tệp hoặc mục tiêu nào trong danh sách phụ thuộc đã thay đổi, ‘make’ sẽ thực hiện các hành động liên quan để cập nhật mục tiêu đó.
5. Build system: Hệ thống xây dựng (build system) là một cách tổ chức và quản lý quá trình biên dịch và xây dựng của một dự án phần mềm. Nó bao gồm các công cụ, quy tắc và quy trình để tự động hóa việc tạo ra các tệp thực thi hoặc thư viện từ mã nguồn.
6. 8bit/16bit/32bit machine

Các thuật ngữ 8-bit, 16-bit và 32-bit machine đề cập đến độ rộng của một từ (word) máy tính, có nghĩa là số bit mà máy tính có thể xử lý trong một lần thực hiện các phép tính và thao tác dữ liệu.

-Máy tính 8 bit:

+Trong máy tính 8-bit, mỗi từ dữ liệu có độ rộng 8 bit.

+Máy tính 8-bit thường được sử dụng trong các thiết bị nhúng, máy tính cá nhân đời đầu như các máy tính Commodore 64, Atari và máy tính trò chơi

+Phạm vi giá trị mà một từ 8-bit có thể biểu diễn là từ 0 đến 255.

-Máy tính 16 bit:

+Trong máy tính 16-bit, mỗi từ dữ liệu có độ rộng 16 bit

+Máy tính 16-bit thường được sử dụng trong các máy tính cá nhân như IBM PC/AT và các hệ thống trò chơi như Sega Genesis

+Phạm vi giá trị mà một từ 16-bit có thể biểu diễn là từ -32,768 đến 32,767 hoặc từ 0 đến 65,535, tùy thuộc vào cách mã hóa

-Máy tính 32 bit

+Trong máy tính 32-bit, mỗi từ dữ liệu có độ rộng 32 bit.

+Máy tính 32-bit đã trở nên phổ biến trong những năm 1990 và được sử dụng rộng rãi trong máy tính cá nhân và máy chủ

+Phạm vi giá trị mà một từ 32-bit có thể biểu diễn là từ -2,147,483,648 đến 2,147,483,647 hoặc từ 0 đến 4,294,967,295, tùy thuộc vào cách mã hóa

1. Tổ chức firmware, kiến trúc phần mềm.

Tổ chức firmware:

-Module chức năng: Phân chia firmware thành các mô-đun chức năng riêng biệt, mỗi mô-đun chịu trách nhiệm cho một chức năng cụ thể của hệ thống nhúng

-Giao diện module: Xác định và thiết kế các giao diện giữa các mô-đun chức năng để chúng có thể tương tác với nhau một cách hiệu quả

-Quản lý tài nguyên: Quản lý tài nguyên phần cứng như bộ nhớ, cổng giao tiếp, ngắt và các tài nguyên khác để đảm bảo rằng chúng được sử dụng một cách hiệu quả

-Kiểm tra và gỡ lối: Xác định và triển khai các công cụ và kỹ thuật để kiểm tra và gỡ lỗi firmware, bao gồm việc sử dụng debuggers, logic analyzers và các công cụ khác

Kiến trúc phần mềm

-Mô hình lớp: Sử dụng mô hình lớp để phân chia chương trình thành các đối tượng độc lập, mỗi đối tượng thực hiện một phần của chức năng hệ thống

-Kiến trúc dựa trên sự kiện: Thiết kế kiến trúc dựa trên sự kiện, trong đó các chức năng được kích hoạt bởi sự kiện nội và ngoại vi

-Kiến trúc hướng dịch vụ: Phân chia chương trình thành các dịch vụ độc lập, mỗi dịch vụ cung cấp một tập hợp các chức năng liên quan

- Kiến trúc lõi: Xác định kiến trúc cốt lõi của hệ thống, bao gồm cấu trúc tổ chức và các quy tắc chung về cách tổ chức và quản lý mã nguồn

-Quản lý tài nguyên: Xác định và triển khai các chiến lược và cơ chế để quản lý tài nguyên phần cứng và phần mềm một cách hiệu quả

Kiến trúc phần mềm và tổ chức firmware đóng vai trò quan trọng trong việc đảm bảo tính linh hoạt, bảo trì và mở rộng của hệ thống nhúng. Việc lựa chọn và triển khai một kiến trúc phù hợp có thể giúp tối ưu hóa hiệu suất và sự ổn định của dự án phần mềm nhúng.

1. Embedded C for 8051.

Tham khảo khóa học.

<https://www.youtube.com/watch?v=Fu-0Yc4ZS-w&list=PLhFjtzzUovr-YW6vlzkiUJRo88T4deV23&ab_channel=H%E1%BB%8CCIT-%C4%90I%E1%BB%86NT%E1%BB%AC>

1. Layout PCB.
2. Tìm hiểu cách vẽ mạch in bằng Altium trên youtube
3. Thực hiện vẽ Kit 8051 với các ngọai vi như: GPIO, LED, BUTTON, UART, khối nạp