1. Ngôn ngữ C là một ngôn ngữ lập trình phổ biến nhất thế giới, là ngôn ngữ đơn giản và linh hoạt khi sử dụng. Nó là một ngôn ngữ lập trình có cấu trúc độc lập và được sử dụng rộng rãi để viết các ứng dụng, hệ điều hành như Windows và nhiều chương trình phức tạp khác như Oracle database, Git, Python Interpreter,…

Ưu điểm:

Gần với ngôn ngữ máy với hiệu suất cao, hỗ trợ trực tiếp cho phần cứng, phổ biến trong lĩnh vực nhúng.

Nhược điểm:

Cú pháp phức tạp, ít hỗ trợ cho lập trình hướng đối tượng và dễ gặp lỗi bảo mật

Phạm vi ứng dụng:

Trong thực tế, ngôn ngữ lập trình C không nằm giới hạn ở việc phát triển hệ điều hành và ứng dụng mà còn được sử dụng trong các lĩnh vực như phát triển giao diện đồ họa người dùng, môi trường phát triển tích hợp,...

2.

**Biến**: Là nơi lưu trữ tạm thời dữ liệu với một kiểu dữ liệu xác định. Biến có thể thay đổi giá trị trong quá trình thực thi chương trình

Trong ngôn ngữ C, một biến sẽ có 3 thành phần:

Kiểu dữ liệu: chính là phần bên trên chúng ta đã tìm hiểu qua.

Tên gọi: tên gọi này sẽ giúp ta làm việc với biến đó trong chương trình, ví dụ gọi biến đó ra, tính toán biến đó...

Giá trị: đó là giá trị mà biến đó nó chứa, ví dụ như 10,5,4…

**Hằng**: là một biến có giá trị không thể thay đổi.

Có 2 cách khai báo hằng:

Cách 1:Sử dụng bộ tiền xử lý #define.

Cách 2: Sử dụng từ khóa const.

**Toán tử trong C:**

Toán tử số học: +,-,\*,/

Toán tử so sánh: >,<,>=,<=,!=,==,

**Toán tử logic**: Trả về giá trị là true ỏ false

Số 0 biểu diễn cho trạng thái false (sai).

Tất cả các số nguyên khác 0 biểu diễn cho trạng thái true (đúng).

&&,||,!.

**Toán tử thao tác trên bit: &,|,~,>>,<<.**

**Toan tử gán:** Sử dụng dấu = cho việc gán giá trị vào biến.

Ví dụ:

int a=5;

**Các toán tử gán mở rộng:**

**Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động**

**Toán tử tăng giảm:**

++ là toán tử tăng

++i tương đương với i=i+1.

--i tương đương với i=i-1.

++i thì i được tăng trước sau đó sẽ lấy kết quả để thực hiện biểu thức

i++ thì i được đưa vào thực hiện biểu thức trước sau đó mới tăng i lên.

**Một số toán tử khác:**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động**

**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 :**

Đây là các kiểu dữ liệu cơ bản trong C:

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động

3 . Cấu trúc điều kiện và vòng lặp trong C

**Điều kiện if- else :**

 Được sử dụng để thực hiện một hành động nào đó nếu điều kiện đúng (true), và một hành động khác nếu điều kiện sai (false). Cú pháp của câu lệnh if-else như sau:

if(điều kiện) {

Code chạy nếu thỏa mãn điều kiện của if

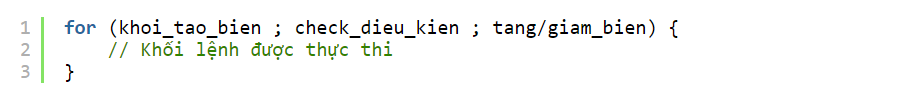
} else {

Code chạy nếu thỏa mãn điều kiện của else

**}**

**Vòng lặp for:**

Được sử dụng để lặp lại một khối lệnh một số lần nhất định. Cú pháp của vòng lặp for như sau:



**Vòng lặp while:**

Cho phép lặp lại một khối mã cho đến khi một điều kiện đã cho là sai. Cú pháp của vòng lặp while như sau:

while(condition) {

statements;

}

**Vòng lặp do-while:**

Cũng giống như vòng lặp while, tuy nhiên, câu lệnh sẽ được thực hiện ít nhất một lần ngay cả khi điều kiện không đúng từ đầu. Cú pháp của vòng lặp do-while như sau:

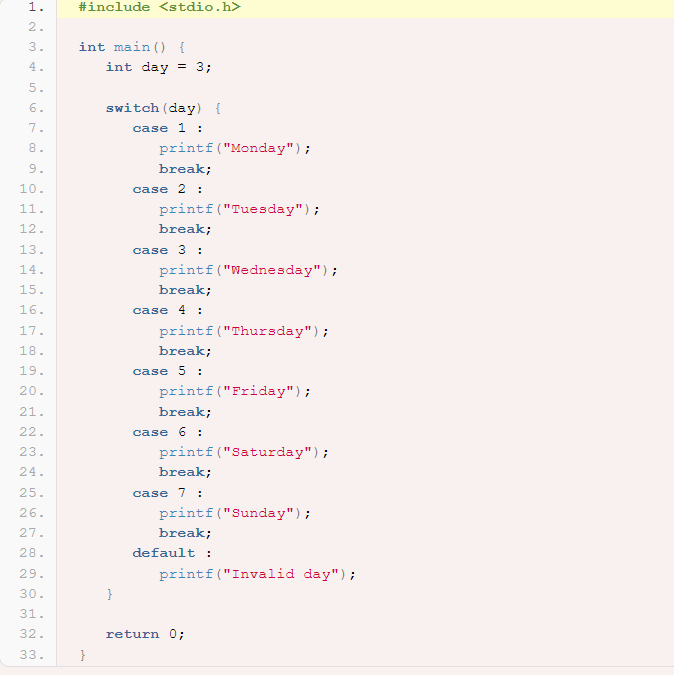
do {

statements;

} while (condition);

**Switch case:**

Cho phép kiểm tra một biến hoặc một biểu thức với nhiều giá trị khác nhau và thực hiện các hành động tương ứng với giá trị của biến hoặc biểu thức đó. Switch case thường được sử dụng khi cần xử lý nhiều trường hợp khác nhau, tùy thuộc vào giá trị của biến hay biểu thức.



4. Các kiểu dữ liệu tự định nghĩa Struct, Union:

**Structure** trong C (cấu trúc trong C) là một kiểu dữ liệu người dùng tự định nghĩa (user defined datatype) cho phép bạn lưu trữ các loại phần tử khác nhau.Mỗi phần tử của một cấu trúc được gọi là một thành viên (member).

Cấu trúc :

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Có hai cách để khai báo biến cấu trúc:

1.Sử dụng từ khóa struct trong hàm main().

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

2.Khai báo biến tại thời điểm định nghĩa cấu trúc.

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, hàng

Mô tả được tạo tự động2

Truy cập các thành viên của cấu trúc

Có hai cách để truy cập vào các thành viên cấu trúc:

Bởi . (thành viên hoặc toán tử chấm).

Bởi -< (toán tử con trỏ cấu trúc).

**UNION:**

Cũng giống như structure, union trong C là kiểu dữ liệu do người dùng định nghĩa được sử dụng để chứa các loại phần tử khác nhau.Nhưng nó không chiếm tổng bộ nhớ của các thành viên cộng lại, mà nó chỉ chiếm bộ nhớ của thành viên có lớn nhất.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

5. Con trỏ, mảng

**Con trỏ**: cũng là một dạng biến có địa chỉ và giá trị đi kèm. Nhưng pointer (con trỏ C) là một biến đặc biệt, nó không lưu trữ những giá trị thông thường. Biến của con trỏ sẽ trỏ đến một địa chỉ của biến khác. Chính vì thế giá trị của nó ở trong dạng một địa chỉ.

Khai báo con trỏ:

<kiểu dữ liệu> \* <tên biến> hoặc

<kiểu dữ liệu>\* <tên biến> hay

<kiểu dữ liệu> \*<tên biến>

Gán địa chỉ vào con trỏ trong C:

TH1: Gán địa chỉ sau khi đã khai báo: Trong ví dụ dưới đây, giá trị được gán cho p là địa chỉ của value.

int \*p, value;

value = 5;

p = &value;

TH2: Khai báo và gán địa chỉ đồng thời: Trong ví dụ dưới, hoạt động khai báo và gán giá trị cho p là địa chỉ của value được thực hiện cùng lúc.

int value = 5;

int \*p = &value;

**Mảng (array) trong C** : là một tập hoặc một nhóm các phần tử (dữ liệu) có kiểu dữ liệu đồng nhất(tương tự). Các phần tử của mảng được lưu trong các vùng nhớ liên tiếp.

Bạn có thể khai báo một mảng trong ngôn ngữ c theo cách sau.

data\_type array\_name[array\_size];

Con trỏ mảng biểu thị địa chỉ của điểm bắt đầu vùng lưu trữ mảng trong bộ nhớ. Địa chỉ này cũng chính là địa chỉ của phần tử đầu tiên (có index bằng 0) của mảng trong bộ nhớ.

Cách truy cập phần tử trong mảng bằng con trỏ: \*(p + <phần tử muốn truy cập>)

VD: a[1] ~ \*(p + 1).

6. Hàm, tham chiếu, tham trị

**Hàm trong C:** là một tập hợp các xử lý nhằm thực hiện một chức năng cụ thể nào đó trong chương trình. Hàm cho phép bạn kết hợp các xử lý khác nhau thành một và đặt tên cho nó. Sau khi tạo và đặt tên cho một hàm, chúng ta chỉ cần gọi tên hàm ra mỗi khi cần sử dụng đến nó trong chương trình.

Cấu trúc hàm gồm 3 thành phần chính, đó là tham số, đối số và giá trị trả về.

Ví dụ:

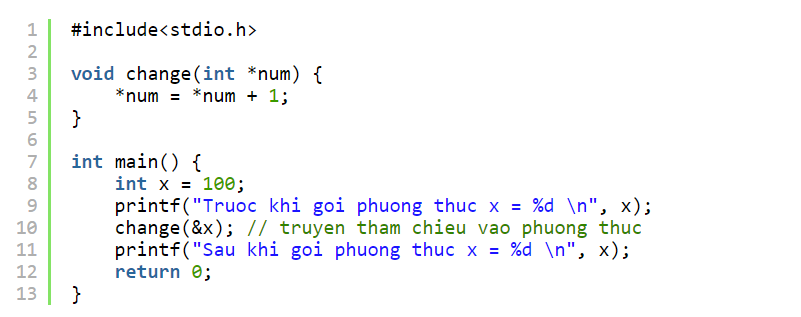
int find\_sum(int a,int b){

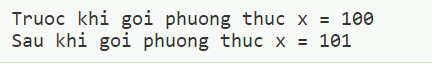
int sum = a + b;

return sum;

}

**Truyền tham chiếu** :là truyền địa chỉ ô nhớ của biến, do đó khi thay đổi giá trị của biến bên trong phương thức thì giá trị của biến cũng bị thay đổi bên ngoài phương thức.





**Truyền tham trị** :là truyền giá trị của biến (không phải là địa chỉ ô nhớ), khi đó phương thức sẽ tự động tạo ra một địa chỉ ô nhớ mới để lưu trữ giá trị này, do đó nó chỉ được thay đổi trong phương thức hiện hành và giá trị của biến không bị thay đổi bên ngoài phương thức hiện hành.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, màu trắng, thuật in máy

Mô tả được tạo tự động

II. Embedded C

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

|  |  |  |
| --- | --- | --- |
| TÍnh năng | C | Embedded C |
| Bộ nhớ | Không dành cho hệ thống với bộ nhớ hạn chế | Dành cho hệ thống với bộ nhớ hạn chế |
| Khả năng xử lí | Không dành cho hệ thống với khả năng xử lí hạn chế | Dành cho hệ thống với khả năng xử lí hạn chế |
| Trình biên dịch (Complier) | Trình biên dịch dùng được cho nhiều nền tảng | Trình biên dịch chỉ dùng được cho một số nền tảng đặc biệt |
| Thư viện | Có thư viện lớn | Thư viện bé hơn |
| Khả năng mở rộng | Có thể mở rộng | Không thể mở rộng |
| Phụ thuộc vào phần cứng | Không | Có |
| Khả năng xử lí tác vụ | Có thể xử lí đa tác vụ cùng một lúc | Không thể xử lí đa tác vụ cùng một lúc |
| Cách lập trình | Lập trình tự do | Lập trình theo cấu trúc |

2.Lưu ý khi lập trình C embedded

Đặc điểm đối với hệ thống nhúng:

ROM và RAM hạn chế. Lập trình phụ thuộc phần cứng. Cần đáp ứng chính xác về thời gian (hàm xử lý ngắt, tác vụ…) Nhiều kiểu pointer (far/rom/ui/paged/…) Một số keywords và token đặc biệt (@, interrupt, tiny,..).

Để phát triển tốt phần mềm nhúng bằng ngôn ngữ C cần nắm vững.

Thiết kế kiến trúc phần mềm hợp lý. Thành thạo sử dụng các tool và debugging Data types native support Các thư viện chuẩn. Phân biệt rõ về simple code với eficient code.

3.C memory management

Ngôn ngữ lập trình C cung cấp một số hàm để cấp phát và quản lý bộ nhớ,có thể tìm thấy các hàm này trong file header <stdlib.h>.

**Hàm malloc()** viết tắt của từ memory allocation tức là cấp phát động vùng nhớ, hàm này được sử dụng để xin cấp phát khối bộ nhớ theo kích thước byte mong muốn.

Giá trị trả về của hàm là một con trỏ kiểu void, bạn nên ép kiểu sang kiểu dữ liệu mà bạn cần dùng.

Các giá trị trong các ô nhớ được cấp phát là giá trị rác

Cú pháp : ptr = (cast\_type\*)malloc(byte\_size)

Trong đó :

ptr là con trỏ lưu trữ ô nhớ đầu tiên của vùng nhớ được cấp phát

cast\_type\* là kiểu con trỏ mà bạn muốn ép kiểu sang

byte\_size là kích thước theo byte bạn muốn cấp phát



**Hàm calloc()** viết tắt của contiguous allocation tương tự như malloc() sử dụng để cấp phát vùng nhớ động nhưng các giá trị của các vùng nhớ được cấp phát sẽ có giá trị mặc định là 0 thay vì giá trị rác như hàm malloc()

Cú pháp : ptr = (cast\_type\*) calloc(n, element\_size)

Trong đó :

ptr là con trỏ lưu trữ ô nhớ đầu tiên của vùng nhớ được cấp phát

cast\_type\* là kiểu con trỏ mà bạn muốn ép kiểu sang

n là số lượng phần tử bạn muốn cấp phát

element\_size là kích thước theo byte của 1 phần tử



Hàm malloc() và calloc() xin cấp phát vùng nhớ nhưng lại không tự giải phóng vùng nhớ mà nó xin cấp phát, hàm free() có chắc năng giải phóng vùng nhớ mà malloc() hoặc calloc() đã xin cấp phát.

Việc sử dụng free() sau khi sử dụng malloc() và calloc() là cần thiết để tránh lãng phí bộ nhớ

Cú pháp : free(ptr)

**Hàm realloc()** viết tắt của re-allocation tức là cấp phát lại, trong trường hợp sử dụng malloc() và calloc() nhưng cần bổ sung thêm bạn sử dụng realloc().

realloc() giúp bạn giữ lại các giá trị trên vùng nhớ cũ và bổ sung thêm vùng nhớ mới với các giá trị rác.

Cú pháp : ptr = (cast\_type\*)realloc(ptr, new\_size)

4. Compilation process, toolchain, compiler, linker.

**Compilation process:**

1. Giai đoạn tiền xử lý – Preprocessor

Giai đoạn này sẽ thực hiện:

Nhận mã nguồn

Xóa bỏ tất cả chú thích, comments của chương trình

Chỉ thị tiền xử lý (bắt đầu bằng #) cũng được xử lý

2. Cộng đoạn dịch Ngôn Ngữ BậcCao sang Assembly

Phân tích cú pháp (syntax) của mã nguồn NNBC

Chuyển chúng sang dạng mã Assembly là một ngôn ngữ bậc thấp (hợp ngữ) gần với tập lệnh của bộ vi xử lý.

3. Công đoạn dịch Assembly

Dich chương trình => Sang mã máy 0 và 1

Một tệp mã máy (.obj) sinh ra trong hệ thống sau đó.

4. Giai đoạn Linker

Trong giai đoạn này mã máy của một chương trình dịch từ nhiều nguồn (file .c hoặc file thư viện .lib) được liên kết lại với nhau để tạo thành chương trình đích duy nhất

Mã máy của các hàm thư viện gọi trong chương trình cũng được đưa vào chương trình cuối trong giai đoạn này.

**Toolchain:**

Toolchain là tập hợp của các tool dùng để biên dịch source code thành các file executable có thể chạy được trên target device (thiết bị thật). Toolchain bao gồm compiler, linker, các thư viện runtime và một số tool linh tinh khác.

Native toolchain: Loại toolchain chạy trên hệ thống giống với target device (device dùng để run chương trình build ra bởi toolchain). Thường là toolchain cho desktop dùng để build ra các chương trình chạy trên destop luôn.

Cross toolchain: Loại toolchain này chạy trên hệ thống khác với target device, ví dụ dùng desktop để build ra chương trình chạy trên một thiết bị nhúng khác. Toolchain này giúp cho quá trình phát triển phần mềm trở nên nhanh hơn vì chúng ta có thể develop phần mềm trên desktop PC và load chúng lên chạy trên target device.

**Compiler:**

Compiler hay còn gọi là trình biên dịch có thể được hiểu là công việc dịch chuỗi câu lệnh được viết từ một ngôn ngữ lập trình thành chương trình tương đương dưới dạng ngôn ngữ máy tính, thường là ngôn ngữ ở cấp thấp hơn, ngôn ngữ máy. Đơn giản dễ hiểu thì có thể tạm nói là nhờ Complier này mà file .c chúng ta viết mới được dịch thành file .hex .bin để nạp được xuống một MCU bất kỳ.

**Linker:**

Linker là công cụ kết hợp các tập tin object và tập tin nén, sắp xếp lại dữ liệu của các tập tin đó và liên kết chúng lại với nhau thành tập tin thực thi.

Linker còn có tạo ra bản đồ liên kết (link map) vào tập tin output chuẩn. Bản đồ này sẽ cung cấp các thông tin về sự ánh xạ của tập tin object được ánh xạ vào bộ nhớ như thế nào, giá trị cấp pháp ra sao…

Nói một cách “nôm na”, Compiler sẽ tạo ra các file mã máy mà không cần biết nó sẽ được bố trí vào bộ nhớ như thế nào. Việc bố trí các file mã máy này sẽ là công việc của Linker.

6. 8 bit/16 bit/ 32 bit

8 bit: Máy 8-bit có khả năng xử lý dữ liệu với các giá trị từ 0 đến 255 (2^8 - 1). Đây là các máy tính cổ điển như các vi xử lý Intel 8080, Zilog Z80, hoặc MOS 6502. Mặc dù ít phổ biến hơn trong các ứng dụng hiện đại, nhưng vẫn được sử dụng trong các hệ thống nhúng đơn giản hoặc trong việc điều khiển các thiết bị cơ bản.

16 bit: Máy 16-bit có khả năng xử lý dữ liệu với các giá trị từ 0 đến 65,535 (2^16 - 1). Các ví dụ bao gồm Intel 8086/8088 và các vi xử lý Motorola 68000. Máy 16-bit thường được sử dụng trong các hệ thống nhúng đòi hỏi tính toán đơn giản đến trung bình.

32 bit: Máy 32-bit có khả năng xử lý dữ liệu với các giá trị từ 0 đến 4,294,967,295 (2^32 - 1). Đây là một tiêu chuẩn phổ biến trong máy tính hiện đại. Các CPU 32-bit như Intel 80386, ARM Cortex-M3 hoặc các phiên bản sau của vi xử lý Motorola 68000 thường được sử dụng trong các hệ thống nhúng phức tạp hơn và các máy tính cá nhân.

7. Tổ chức firmware, kiến trúc phần mềm.

Firmware được cấu thành từ hai những thành khác biệt, gồm hai phần lớn là BIOS và EFI. Firmware có chức năng gần giống với Drive, là bộ phận giao thao giữ phần cứng và phần mềm:

Đối với Bios, sau khi nhấn nút để mở nguồn máy tính thì máy tính sẽ khởi động vào Bios sẽ được kiểm tra tất cả các lỗi nhờ sự tương tác ngay và luôn với phần cứng. Bios còn là chỉ huy trưởng cho một chương trình có tên gọi khác là Bootloader thực hiện nhiệm vụ đánh thức hệ điều hành trên máy tính đang ngủ ngon trong ổ cứng và đưa nó đến bộ nhớ dữ liệu tạm thời.Nhiệm vụ của Bios là đảm bảo các thành phần máy tính hoạt động đúng cách và xử lý các thành phần của phần cứng máy tính. Do là phần mềm cấp thấp và hầu như không có sự thay đổi, Bios trở nên lỗi thời và không hỗ trợ các công nghệ hiện đại.

Efi là một bộ đặc tả giao thức phần mềm chịu trách nhiệm giao tiếp giữa hệ điều hành và hệ thống này được dùng để khởi động phần cứng và bỏ qua bootloader Nó còn được gọi với cái tên khác Uefi và có ưu điểm hơn so với Bios. Efi hỗ trợ tính năng Secure boot để cải thiện tính bảo mật, đảm bảo máy tính của bạn chỉ khởi động bằng phần mềm được sản xuất máy tính tin cậy tránh khả năng bị đánh mất hoặc thay thế

Kiến trúc phần mềm:

Phần mềm hệ thống nhúng là các chương trình được viết, thiết kế, lập trình chuyên biệt cho các thiết bị số và hoạt động song song với các thiết bị đó. Hiện nay có 6 loại kiến trúc phần hệ thống nhúng phổ biến là:

Vòng lặp kiểm soát đơn giản:

Một vòng lặp kiểm soát là một hệ thống quản lý quá trình được thiết kế để duy trì một quá trình diễn biến tại một vị trí mong muốn.

Hệ thống ngắt điều khiển:

Hệ thống nhúng thường được điều khiển bằng các ngắt, các tác vụ của hệ thống nhúng được kích hoạt bởi các loại sự kiện khác nhau.

Đa nhiệm tương tác:

Hệ thống đa nhiệm không ưu tiên gần giống với kỹ thuật của vòng lặp kiểm soát đơn giản, chỉ khác ở chỗ trừ việc vòng lặp này được ẩn giấu thông qua một giao diện lập trình API.

Đa nhiệm ưu tiên:

Hệ thống đa nhiệm ưu tiên thường có một đoạn mã ở mức thấp thực hiện việc chuyển đổi giữa các tác vụ khác nhau thông qua một bộ định thời.

Vi nhân (Microkernel) và nhân ngoại (Exokernel):

Vi nhân (Microkernet) là một bước tiếp cận gần hơn tới khái niệm hệ điều hành thời gian thực. Lúc này, nhân hệ điều hành thực hiện việc cấp phát bộ nhớ và chuyển CPU cho các luồng thực thi.

Nhân khối (monolithic kernels):

Nhân ngoại (Exokernel) là tiến hành giao tiếp hiệu quả bằng cách sử dụng các lời gọi chương trình con thông thường.