1. Tìm hiểu về lập trình C cơ bản:
2. Ưu nhược điểm của C:

1.1 Ưu điểm:

C là khối xây dựng cho những ngôn ngữ lập trình khác

Ngôn ngữ mạnh mẽ, hiệu quả

Linh hoạt

Có thể cấp phát bộ nhớ động

* 1. Nhược điểm:

Không hỗ trợ OOP (Kế thừa, đa hình, đóng gói, trừu tượng, ẩn dữ liệu)

Không có hàm tạo hoặc hàm hủy

Không có tính năng namespace (không thể sử dụng lại tên biến trong cùng 1 phạm vi)

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

BIẾN:

Biến 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, do đó được gọi là biến.

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

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

Có 2 cách khai báo hằng

1. #define
2. const

TOÁN TỬ:

Toán tử số học: +, -, \*, /

Toán tử so sánh: >, <, ==, !=, >=, <=

Toán tử logic: %% (and), || (or), ! (not)

Toán tử thao tác trên bit: % (and bit), | (or bit), ~ (phủ định), >> (dịch bit sang phải), << (dịch bit sang trái)

Toán tử gán: +=, -=, \*=, /=, %=. <<=, >>=, &=, |=

Toán tử tang giảm: ++ (tăng), -- (giảm)

i++ (tăng sau): lấy giá trị của i để thực hiện biểu thức xong mới tăng i.

++i (tăng trước): tăng giá trị của i trước xong mới thực hiện biểu thức.

Một số toán tử khác:

sizeof(): trả về kích cỡ một biến

&: trả về địa chỉ một biến

\*: trỏ tới một biến

3. Các cấu trúc điều kiện, vòng lặp

CẤU TRÚC ĐIỀU KIỆN:

Cấu trúc if()

if(điều kiện) {

Code chạy nếu thỏa mãn điều kiện của if

} else {

Code chạy nếu thỏa mãn điều kiện của else

}

Cấu trúc switch()

Switch(điều kiện){

case (tên của case):

Code…

break; (để kết thúc 1 case trong switch)

.

.

.

default: (lệnh của default sẽ chạy khi không có case nào thỏa mãn)

Code…

CẤU TRÚC VÒNG LẶP:

Cấu trúc while()

while(condition) {

statements;

}

Cấu trúc do while (chạy code trong do trước rồi mới xét điều kiện của while(nếu true 🡪 chạy tiếp | false 🡪 dừng)

do {

statements;

} while (condition);

Vòng lặp for():

for(Khởi tạo giá trị biến lặp, điều kiện biến lặp, cập nhật giá trị biến lặp) {

statements;

}

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

STRUCT:

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).

struct structure name {

data\_type member1;

data\_type member2;

…

}

Có hai cách để khai báo biến cấu trúc:

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

struct structname tên biến 1,…;

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

struct structure name {

data\_type member1;

data\_type member2;

…

} tên biến 1, ….;

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.

Cú pháp như struct

5. Con trỏ, mảng

CON TRỎ:

Con trỏ C 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>;

MẢNG:

Mảng là một loại cấu trúc dữ liệu trong [ngôn ngữ lập trình C/C++](https://quantrimang.com/hoc/cplusplus), nó lưu trữ một tập hợp tuần tự các phần tử cùng kiểu với độ dài cố định. Mảng thường được sử dụng để lưu trữ tập hợp dữ liệu, nhưng nó cũng hữu dụng khi dùng để lưu trữ một tập hợp biến có cùng kiểu.

Khai báo mảng: <kiểu dữ liệu> <tên mảng> [Kích cỡ mảng];

Khi khai báo một mảng, tên biến dùng để khai báo mảng sẽ biểu thị địa chỉ của điểm bắt đầu của vùng lưu mảng trong bộ nhớ.

Do tên biến dùng để khai báo mảng đã là một địa chỉ, nên khi gán địa chỉ của mảng cho con trỏ, chúng ta không cần thêm toán tử & như các loại dữ liệu khác.  
Ngược lại, nếu bạn viết thêm toán tử & thì khi biên dịch sẽ xảy ra lỗi.

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 (function):

Hàm (function) là một các khối lệnh có nhiệm vụ thực hiện một chức năng nào đó.

Data\_type function\_name(type1 parameter 1, type2 parameter2) {

//code

}

Cách gọi hàm: <tên hàm>()

Tham số (Parameter) hay tham số hình thức là các thành phần khi xây dựng hàm.

Đối số (Argument) hay tham số chính thức là các giá trị truyền vào cho hàm khi gọi hàm.

Khi hàm của thực hiện các chức năng tính toán và mong muốn trả về 1 giá trị cụ thể thì cần câu lệnh return

THAM CHIẾU VÀ THAM TRỊ

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.

1. Embedded C
2. Sự giống và khác nhau giữa Embedded C và C

Giống nhau: một số syntax, hardware, software

Khác nhau:

|  |  |  |
| --- | --- | --- |
| 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 |

1. Lưu ý cho embedded C:

Phân biệt rõ simple code và efficient code

Cần đáp ứng chính xác về thời gian

Nhiều kiểu con trỏ

1. C memory management

Ngôn ngữ lập trình C cung cấp một số hàm để cấp phát và quản lý bộ nhớ. Các hàm này được cài đặt trong tệp tiêu đề <stdlib.h>.

|  |  |
| --- | --- |
| Hàm | Chức năng |
| malloc() | Phân bổ kích thước yêu cầu của byte và trả về một byte đầu tiên của không gian được phân bổ. |
| calloc() | Phân bổ không gian cho một phần tử mảng, khởi tạo về 0 và sau đó trả về một con trỏ tới bộ nhớ. |
| free() | Giải phóng không gian được phân bổ trước đó. |
| realloc() | Thay đổi kích thước của không gian được phân bổ trước đó. |

1. Compilation progress, toolchain, complier, linker.

COMPILATION PROGRESS:

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

Ví dụ: chỉ thị #include cho phép ghép thêm mã chương trình của một tệp tiêu để vào mã nguồn cần dịch. Các hằng số được định nghĩa bằng #define sẽ được thay thế bằng giá trị cụ thể tại mỗi nơi sử dụng trong chương trình.

2. Cộng đoạn dịch Ngôn Ngữ Bậc Cao 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.

Chính vì vậy mà các lỗi liên quan đến việc gọi hàm hay sử dụng biến tổng thể mà không tồn tại sẽ bị phát hiện. Kể cả lỗi viết chương trình chính không có hàm main() cũng được phát hiện trong liên kết.

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 khác.

Toolchain luôn có ít nhất 3 thành phần sau:

1. Binutils: là tập hợp các binary tool như as (assembler), ld (linker), ar, objcopy, …
2. Compiler: C compiler và C++ compiler
3. C library: là một bộ các API chuẩn dựa trên POSIX, các ứng dụng sử dụng C library để giao tiếp với kernel

Toolchain có 2 loại: Naïve toolchain và Cross toolchain.

**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.

**COMPLIER**

Compiler (trình biên dịch) là một chương trình máy tính làm công việc dịch các chuỗi câu lệnh viết bằng ngôn ngữ lập trình thành chương trình tương đương nhưng viết dưới dạng ngôn ngữ máy tính. Chương trình mới này được gọi là mã đối tượng (object code).

**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.