# LẬP TRÌNH C CƠ BẢN

# A. PHẦN CƠ BẢN

## I. TỔNG QUAN VỀ CHƯƠNG TRÌNH C

### 1. Giới thiệu chương trình cơ bản.

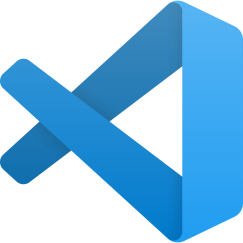
- Các chương trình thường gặp trong ngày.

+ Hệ điều hành:





+ Các ứng dụng:



- Nhìn sơ qua thì chúng ta sẽ thấy từ **chương trình** đến **hệ điều hành điều** có một nguyên lý cơ bản là như sau:  
 + **Phần chương trình chính** : Luôn có một nhân để xử lý trong một hệ thống.

+ **Phần tài nguyên có sẵn**: Là phần tài nguyên có sẵn để có thể xử lý khi nhận dữ liệu đầu vào:

+ **Phần dữ liệu đầu vào**: Dữ liệu mà ta muốn xử lý chúng.  
 + **Phần thông tin** :Phần thông tin đưa ra sau khi sử dụng dữ liệu.

Tài nguyên sẵn có

Thông tin

Dữ liệu

*Hình 1. Cơ bản của chương trình*

- Một ví dụ cơ bản là bạn giải một diện tích hình chữ nhật với **chiều dài = 15**, **chiều rộng = 10**. Thì bạn chắc **phải có** công thức tính trong đầu là **S = chiều dài x chiều rộng**.

Dữ liệu là chiều dài = 15, chiều rộng = 10. Cùng tài nguyên sẳn có ( công thức tính) , bạn sẽ đưa ra thông tin, diện tích là 150. Nguyên lý hoạt động của máy tính cơ bản là như vậy, sẽ có đầu vào, đầu ra, phần xử lý, thông tin.

### 2. Lập trình máy tính căn bản là gì?

**Lập trình máy tính** hay **lập chương trình máy tính** thường gọi tắt là lập trình ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): **Computer programming**, thường gọi tắt là **programming**) là việc lập ra [chương trình làm việc cho máy](https://vi.wikipedia.org/wiki/Ch%C6%B0%C6%A1ng_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh) có [bộ xử lý](https://vi.wikipedia.org/wiki/CPU), nói riêng là [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh), để thực thi nhiệm vụ xử lý thông tin nào đó. Chương trình này gồm dãy các chỉ thị (hay lệnh) hợp lý để máy thực hiện theo trình tự thời gian.

*(Nguồn: https://vi.wikipedia.org/wiki/Lập\_trình\_máy\_tính)*

Một cách dễ hiểu nữa, Lập trình máy tính chính là chúng ta sử dụng các công cụ cần thiết để giao tiếp với máy tính ( ra lệnh, yêu cầu, nhờ vả) máy tính để chúng thực hiện công việc thay cho chúng ta, với một kịch bản soạn thảo trước đó.

### 3. Lập trình c là gì?

- Lập trình ngôn ngữ C chính là dùng ngôn ngữ c để lập trình máy tính.

- **C là một ngôn ngữ lập trình cấp trung**, ngôn ngữ lập trình C được phát triển bởi **Dennis M. Ritchie** để phát triển hệ điều hành UNIX tại Bell Labs. C được thực thi lần đầu tiên trên máy tính DEC PDP-11 vào năm 1972.

- Được coi là ngôn ngữ mẹ của các ngôn ngữ khác, là nền tảng của các ngôn ngữ khác.

- Điều kiện tiên quyết học ngôn ngữ c hay lập trình tốt hơn:  
 + Hiểu biết ngôn ngữ lập trình.

+ Biết được ưu nhược điểm của c.

+ Trước khi học lập trình, bạn phải có kiến thức căn bản về máy tính.

Viết chương trình trên IDE

Code được chuyển sang dạng mã máy để máy tính đọc hiểu và thực thi

Những dòng code thường thấy

Máy tính thực thi

*Hình 2: Sơ đồ máy tính thực thi từ coding đến running*

*- Chương trình c biên dịch và thực thi như thế nào?*

Quy trình dịch là quá trình chuyển đổi từ ngôn ngữ bậc cao (NNBC) (C/C++, Pascal, Java, C#…) sang ngôn ngữ đích (ngôn ngữ máy) để máy tính có thể hiểu và thực thi. Ngôn ngữ lập trình C là một ngôn ngữ dạng biên dịch. Chương trình được viết bằng C muốn chạy được trên máy tính phải trải qua một quá trình biên dịch để chuyển đổi từ dạng mã nguồn sang chương trình dạng mã thực thi. Quá trình được chia ra làm 4 giai đoạn chính:

- Giai đoàn tiền xử lý (Pre-processor)

- Giai đoạn dịch NNBC sang Asembly (Compiler)

- Giai đoạn dịch asembly sang ngôn ngữ máy (Asember)

- Giai đoạn liên kết (Linker)



*Hình 2.1 Quá trình biên dịch*

**\*** HOẠT ĐỘNG

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.

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.

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.

Giai đoạn Linker

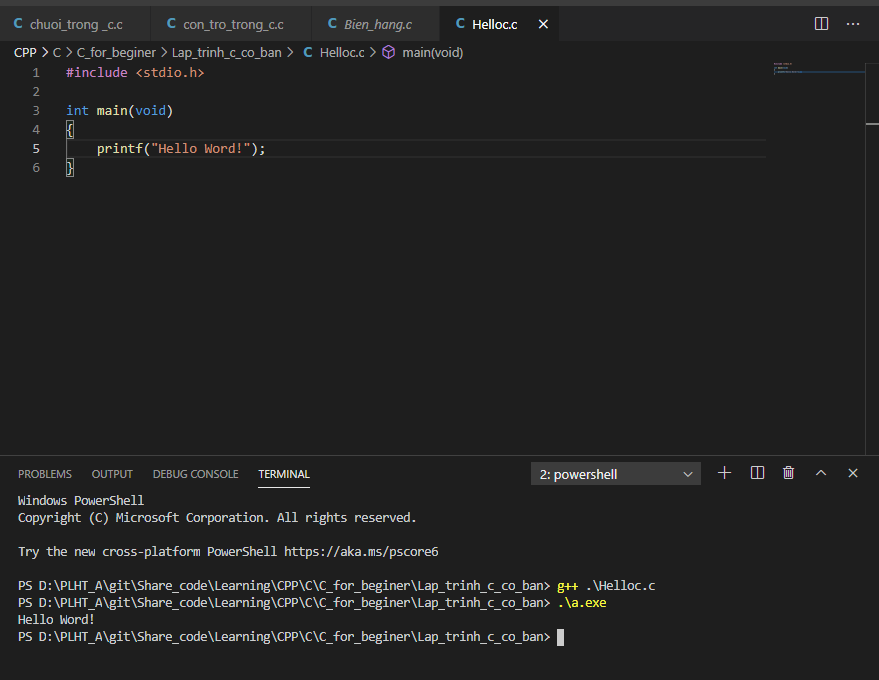
* Trong giai đoạn này mã máy của một chương trình dịch từ nhiều nguồn (file .c hoặc file thư viện .lib) được liên kết lại với nhau để tạo thành chương trình đích duy nhất
* Mã máy của các hàm thư viện gọi trong chương trình cũng được đưa vào chương trình cuối trong giai đoạn này.
* Chính vì vậy mà các 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.

-> Kết thúc quá trình tất cả các đối tượng được liên kết lại với nhau thành một chương trình có thể thực thi được (executable hay .exe) thống nhất.

Biên dịch code thành file .exe trên windows

Chạy file .exe để kiểm thử

Phần code mà chúng ta thường hay viết

*Hình 3. Giới thiệu chương trình c*

## TỔNG KẾT

- Vậy một chương trình từ hệ điều hành đến chương trình con điều gồm những phần cơ bản sao:  
 + Phần yêu cầu dữ liệu cần được xử lý.

+ Phần tài nguyên hiện có.

+ Phần NHÂN để xử lý ( não bộ).

+ Phần đưa ra thông tin.

- Chương trình C cũng có những phần tương tự.  
 + Phần hỗ trợ mà trình biên dịch cung cấp ( thư viện, lệnh tiền xử lý, kiểu dữ liệu, hàm), tôi gọi đây là phần tài nguyên sẳn có.

+ Phần yêu cầu đề bài cần giải quyết, tôi gọi đây là phần yêu cầu dữ liệu cần được xử lý.

+ Trình biên dịch, đây là phần mà chúng ta build hay debug một source bằng g++, g++ sẽ thu thập toàn bộ code, thư viện của chúng ta để biên dịch thành một bộ mã mà máy tính có thể hiểu.  
 + File .exe được tạo (nếu bạn đang sài hệ điều hành windows), đây chính là phần đưa ra thông tin, thông tin chính là một gói chương trình được đưa ra.

- Khi chúng ta soạn thảo Code chương trình C cũng gồm 4 phần:

+ Phần yêu cầu bài toán ( Đây là phần chúng ta cần phải xử lý dữ liệu.

+ Phần tiền xử lý ( Đây là nơi ta khai báo thư viện, tiền lệnh, … là chúng ta đang cung cấp tài nguyên cho chương trình).

+ Phần nhân ( hàm main) phần lỗi của chương trình, xử lý tất cả những phần xử lý.

+ Hàm và biến ( Đây được coi như phần sử dụng dữ liệu thành kiểu dữ liệu có sẳn, xử lý những phần nhỏ để phần main có thể tổng hợp xử lý chung.

+ Phần đưa ra thông tin (Nơi đây chúng ta dùng comment để cho biết đoạn code mà chúng ta code nó thực hiện chức năng gì? Đưa ra thông tin để người trong nhóm có thể nắm bắt được.

Tóm lại, ở đây các bạn cần lưu lý, Một chương trình cơ bản của máy tính; Đặc biệt là phân biệt rõ lập trình c (Từ soạn code đến thành file .exe) và lập trình c ( Quá trình soạn code). Chúng điều có 4 phần nhưng ở mỗi gian đoạn, 4 phần thực hiện ở 1 lĩnh vực khác.

## II. CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH C

### 1. Tiền xử lý.

**Bộ tiền xử lý trong C** ở đây không phải là một phần của bộ biên dịch, nhưng có những bước riêng rẽ trong quá trình biên dịch. Theo cách hiểu cơ bản nhất, bộ tiền xử lý trong ngôn ngữ C là các công cụ thay thế văn bản và hướng dẫn trình biên dịch không yêu cầu tiền xử lý trước khi được biên dịch. Chúng tôi hướng đến bộ tiền xử lý C như CPP.

Tất cả các lệnh tiền xử lý bắt đầu với ký thự “**#”**. Nó ít nhất không phải là ký tự trắng, để dễ dàng đọc.

Ví dụ minh họa:

Phân tích các ví dụ sau để hiểu các directive đa dạng.

#define DO\_DAI\_MANG\_TOI\_DA 20

Tiền xử lý này thông báo cho trình biên dịch C thay thế DO\_DAI\_MANG\_TOI\_DA với 20. Sử dụng *#define* cho các hằng số làm tăng khả năng đọc của chương trình.

#include <stdio.h>

#include "header.h"

Tiền xử lý này thông báo cho trình biên dịch lấy thư viện stdio.h từ **Thư viện hệ thống** và thêm vào mã nguồn hiện tại. Dòng kế tiếp thông báo cho trình biên dịch lấy tệp **header.h** từ thư mục máy tính và thêm nội dung và mã nguồn hiện tại.

#undef KICH\_CO\_FILE

#define KICH\_CO\_FILE 42

Tiền xử lý này thông báo cho trình biên dịch vộ hiệu hóa biến KICH\_CO\_FILE và định nghĩa mới có giá trị 42.

## Dưới đây là danh sách các thẻ tiền xử lý quan trọng trong C:

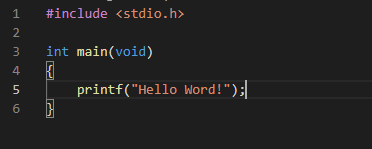
|  |  |
| --- | --- |
| **Directive** | **Miêu tả** |
| #define | Thay thể cho bộ tiền xử lý macro |
| #include | Chèn một header đặc biệt từ file khác |
| #undef | Không định nghĩa một macro tiền xử lý |
| #ifdef | Trả về giá trị true nếu macro này được định nghĩa |
| #ifndef | Trả về giá trị true nếu macro này không được định nghĩa |
| #if | Kiểm tra nếu điều kiện biên dịch là đúng |
| #else | Phần thay thế cho #if |
| #elif | #else một #if trong một lệnh |
| #endif | Kết thúc điều kiện tiền xử lý |
| #error | In thông báo lỗi trên stderr |
| #pragma | Thông báo các lệnh đặc biệt đến bộ biên dịch, sử dụng một phương thức được tiêu chuẩn hóa |

*Bảng 1.Danh sách thẻ tiền xử lý thông dụng*

### 2. Hàm main.

***Hàm main*** là nơi chứa những đoạn code sẽ được chạy đầu tiên, nghĩa là khi biên dịch chương trình thì nội dung trong hàm main sẽ được chạy đầu tiên mà không quan trọng vị trí của nó trong file.

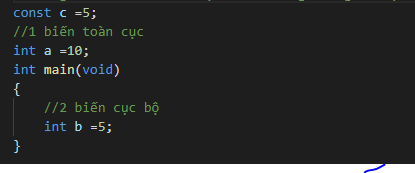
Một cách dễ hiểu hàm main như là một ông nhạc trưởng đang điều khiển ban nhạc vậy, hay một đạo diễn đang quay một bộ phim. Trong Hàm main() ta có thể sắp xếp lại thứ tự tuần tự thực thi của code, cho phép thực thi hay không thực thi hàm và biến đã được khai báo trước đó. Tức là kịch bản viết có diễn viên B,diễn viên B cũng có mặt tại trường quay, nhưng xuất hiện tại ống kính hay không do đạo diễn quyết định. Hàm main cũng tương tự vậy.



*Hình 3.1 Hàm main*

### 3. Biến và hàm khác.

*a. Biến*

Một **biến trong C** là tên của vị trí bộ nhớ. Nó được sử dụng để lưu trữ dữ liệu. Giá trị của nó có thể được thay đổi và nó có thể được sử dụng lại nhiều lần. Mỗi biến trong C có một loại dữ liệu cụ thể, xác định kích thước của bộ nhớ của biến; phạm vi các giá trị có thể được lưu trữ trong bộ nhớ đó.

*Hình 3.2 Biến*

b. Hàm

Hàm trong c là phần các tập lệnh hoặc biến được tạo ra để có thể tái sử dụng, hay thực hiện một chức năng nào đấy mà không cần code thẳng trong hàm main. Hàm main() cũng được gọi là hàm.

### 4. Comments trong C.

Trong lập trình, **Comment** là một dòng hoặc nhiều dòng văn bản, được chèn vào source code chương trình, nhằm làm cho source code trở nên dễ hiểu hơn với người đọc, được bỏ qua bởi **compiler** và **interpreter**.

Trong lập trình comments đối với máy tính không quan trọng vì có hay không có chương trình vẫn thực thi, nhưng đối với lập trình viên thì nó quan trọng vì nó là phần ghi chú cho những lập trình viên có thể biết họ đang code gì, người khác đang làm gì. Đặc biệt trong thời đại hiện tại luôn chú trong làm việc nhóm.

## TỔNG KẾT

Luôn luôn lúc nào trong các phần tôi điều làm các phần sum – phần tổng kết lại các ý trên để chúng ta có cái nhìn tổng quan hơn về phần đã đọc. Trong phần này, ta sẽ hiểu sâu hơn về cấu trúc của một chương trình c căn bản.Đó là :

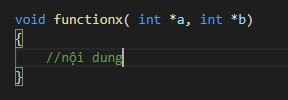
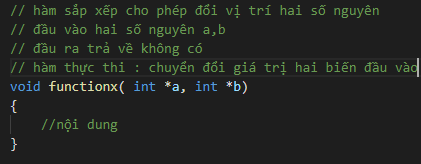
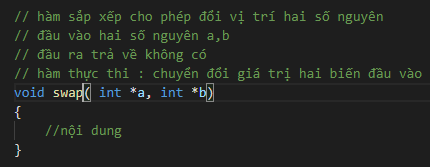
- Tiền xử lý – là phần khai báo tài nguyên được sử dụng.

- Hàm main – đầu não của cả chương trình.

- Biến và hàm – diễn viên và kịch bản có trong chương trình.

- Comments – phần tưởng như là không cần thiết nhưng cực kì quan trọng.

**Mẹo:**

Để phân biệt lập trình viên tiềm năng và các lập trình viên khác, người ta luôn dựa vào việc lập trình viên đấy viết biến, comments như thế nào và cách sắp xếp chúng ra sao.  
ví du:

*Hình 3.2 Các cách diễn đạt*

Thoạt nhìn hình thì ta thấy về mặt nội dung thì cả 3 điều đảm bảo và máy tính vẫn build được hàm, chức năng không thay đổi, nhưng từ trái sang phải thì việc đọc code và hiểu hàm dễ dàng hơn. Chỉ cần nhìn vào thì sẽ thấy ai có tìm năng phát triển bản thân, làm việc nhóm cao hơn. Như người xưa có câu: “Cao thủ so chiêu, thì 1 chiêu phân cao thấp”. Một đoạn mã nguồn là biết tìm lực tới đâu.

Nên mới bắt đầu lập trình đừng nên lười nhé, vì bước đầu luôn là những bước khó đi nhất trong sự nghiệp của mỗi người. Thiên tài và người thường chỉ khác biệt họ cố gắng ở những bước đầu ra sau.

## III. PHẦN HEADER – TIỀN XỬ LÝ

Nhắc lại một số kiến thức về tiền xử lý mà đã được giới thiệu tại mục II phần 1.  
***Bộ tiền xử lý trong C****ở đây không phải là một phần của bộ biên dịch, nhưng có những bước riêng rẽ trong quá trình biên dịch. Theo cách hiểu cơ bản nhất, bộ tiền xử lý trong ngôn ngữ C là các công cụ thay thế văn bản và hướng dẫn trình biên dịch không yêu cầu tiền xử lý trước khi được biên dịch. Chúng tôi hướng đến bộ tiền xử lý C như CPP.*

Tôi đánh dấu nó vì đây chỉ là kiến thức nhắc lại, bạn có thể hiểu cơ bản tiền xử lý nó như kiểu: là phải xử lý trước khi trình biên dịch chạy vào hàm main vậy, nếu hàm main được ví như đạo diễn, thì tiền xử lý là đạo cụ, chuyên chuẩn bị đồ sẵn trước khi ông đạo diễn ngồi xuống ghế và nói “action”.

Chúng ta có 2 dạng tiền xử lý:  
- Lệnh tiền xử lý – tức là đồ mà ông đạo cụ sắp xếp trước.

- Thư viện – tức là dụng cụ, diễn viên, những đạo cụ đã chuẩn bị chờ gọi tên hay sắp xếp.

Một cách dễ hiểu hơn là tiền xử lý chia làm 2 phần:

- Phần được viết trong một file sau đó được khai bao ở chương trình của chúng ta, như file stdio.h thường hay khai báo.

Phần chúng ta viết ngay trên chương trình được khai báo bởi dấu “#” đúng trước.

### 1. Lệnh tiền xử lý (Directive)- Những phần được Ông đạo cụ sắp xếp

Lệnh tiền xử lý trong ngôn ngữ C là các công cụ thay thế văn bản và hướng dẫn trình biên dịch không yêu cầu tiền xử lý trước khi được biên dịch

* Chỉ thị định nghĩa cho tên (#define macro).
* Chỉ thị biên dịch có điều kiện (#if, #else, #elif, #endif, …).

***-* Chỉ thị tiền xử lý *#define*** *tạo các biểu tượng hằng. Biểu tượng hằng là một macro và mẫu chung của chỉ thị tiền xử lý này trong C++ là:*

#define ten\_cua\_macro ten\_thay\_the

*Khi dòng này xuất hiện trong một file, tất cả macro xuất hiện theo sau trong file này sẽ được thay thế bởi ten\_thay\_the trước khi chương trình được biên dịch*

*-* **Chỉ thị biên dịch có điều kiện***(#if, #else, #elif, #endif, …),Có một số chỉ thị tiền xử lý có thể sử dụng để biên dịch có sự tuyển chọn giữa các phần trong source code của bạn.*

*Chỉ lệnh tiền xử lý có điều kiện khá giống với cấu trúc lựa chọn if.*

#ifdef tên\_hàm\_điều\_kiện nội\_dung\_thực\_thi #endif

|  |  |
| --- | --- |
| **Directive** | **Miêu tả** |
| #define | Thay thể cho bộ tiền xử lý macro |
| #include | Chèn một header đặc biệt từ file khác |
| #undef | Không định nghĩa một macro tiền xử lý |
| #ifdef | Trả về giá trị true nếu macro này được định nghĩa |
| #ifndef | Trả về giá trị true nếu macro này không được định nghĩa |
| #if | Kiểm tra nếu điều kiện biên dịch là đúng |
| #else | Phần thay thế cho #if |
| #elif | #else một #if trong một lệnh |
| #endif | Kết thúc điều kiện tiền xử lý |
| #error | In thông báo lỗi trên stderr |
| #pragma | Thông báo các lệnh đặc biệt đến bộ biên dịch, sử dụng một phương thức được tiêu chuẩn hóa |

*Hình 3.3 danh sách các thẻ tiền xử lý quan trọng*

|  |  |
| --- | --- |
| **Macro** | **Miêu tả** |
| \_\_DATE\_\_ | Ngày hiện tại, như là một hằng số ký tự, trong định dạng "MMM DD YYYY" |
| \_\_TIME\_\_ | Thời gian hiện tại, như là một hằng số ký tự, trong định dạng "HH:MM:SS" |
| \_\_FILE\_\_ | Nó chứa tên file hiện tại như là một hằng số chuỗi |
| \_\_LINE\_\_ | Nó chứa số dòng hiện tại như là một hằng số thập phân |
| \_\_STDC\_\_ | Được định nghĩa là 1 khi bộ biên dịch biên dịch với chuẩn ANSI |

## Hình 3.5 Macro được định nghĩa trước trong C

Ta có ví dụ sau:

#include <stdio.h>

#include <stdarg.h>

#include"debug\_macro.h"//

#define MAX\_LINK\_LENGTH (100)

#define MAX\_NUM\_LINK (20)

#define N\_DUMMY\_LINKS (5)

#define N\_DUMMY\_RESULT\_SITE (2)

//

// Dummy danh sách links kết quả tìm kiếm từ google

//

const char\* DUMMY\_RESULT\_LINKS[] =

{

#include"resultLinks.txt"

};

//

// Dummy những site có sample source

//

const char\* DUMMY\_HAS\_SOURCE\_CODE\_SITE[]=

{

"www.stackoverflow.com",

"www.kipalog.com"

};

//

// Hàm tìm kiếm từ khóa trên google, trả về kết quả là danh sách link

//

void search\_google(char\* keyword, char result[MAX\_NUM\_LINK][MAX\_LINK\_LENGTH], int \*nResults)

{

int i = 0;

log\_debug("Search google : (keyword=%s) START \n", keyword);

for (i = 0; i < N\_DUMMY\_LINKS; i++)

{

strcpy(result[i], DUMMY\_RESULT\_LINKS[i]);

}

\*nResults = N\_DUMMY\_LINKS;

log\_debug("Search google : (nResults=%d) END \n", \*nResults);

}

//

// Hàm chạy source sample từ một link đưa vào từ tham số

//

int runSource(char\* link)

{

int retVal = 1;

//log\_debug("Run source in (link=%s) START \n",link);

if (strstr(link, "www.kipalog.com") != NULL)

{

log\_debug("OK, It works!!!!\n");

retVal = 1;

}

else

{

log\_err("Hmm, Bad source (link=%s) \n",link);

retVal = 0;

}

//log\_debug("Run source in (link=%s) END \n",link);

return retVal;

}

//

// Hàm kiểm tra một link có chưa sample source hay không.

//

int hasSampleSource(char\* link)

{

int i = 0;

int retVal = 0;

//log\_debug("Check source (link=%s) START \n",link);

for (i = 0; i < N\_DUMMY\_RESULT\_SITE; i++)

{

if (strstr(link, DUMMY\_HAS\_SOURCE\_CODE\_SITE[i]) != NULL)

{

retVal = 1;

break;

}

}

//log\_debug("Check source (link=%s) END \n",link);

return retVal;

}

//

// Hàm start-up

//

int main(int argc, char\* argv[])

{

char \*keyword;

char results[MAX\_NUM\_LINK][MAX\_LINK\_LENGTH];

int nResults;

int i = 0;

// Kiểm tra tham số

if (2 != argc)

{

log\_err("Invalid parameter, usage: %s <keyword>\n",argv[0]);

return 0;

}

// Lấy keyword

keyword = argv[1];

// Tìm google

search\_google(&keyword[0], results, &nResults);

if (nResults > 0)// Nếu có kết quả

{

// Duyệt từng link kết quả

for (i = 0; i < nResults; i++)

{

log\_debug("Try (link=%s) \n",results[i]);

if (hasSampleSource(results[i]))// Kiểm tra xem có sample source hay không

{

// Chạy thử source sample

runSource(results[i]);

}

else

{

log\_err("No source in link :%s\n", results[i]);

}

}

}

else{

log\_err("Keyword is so difficult (keyword=%s) \n", keyword);

}

return 0;

}

Hoặc ví dụ sau:

#include <stdio.h>

Int main()

{

printf("File :%s\n", \_\_FILE\_\_ );

printf("Date :%s\n", \_\_DATE\_\_ );

printf("Time :%s\n", \_\_TIME\_\_ );

printf("Line :%d\n", \_\_LINE\_\_ );

printf("ANSI :%d\n", \_\_STDC\_\_ );

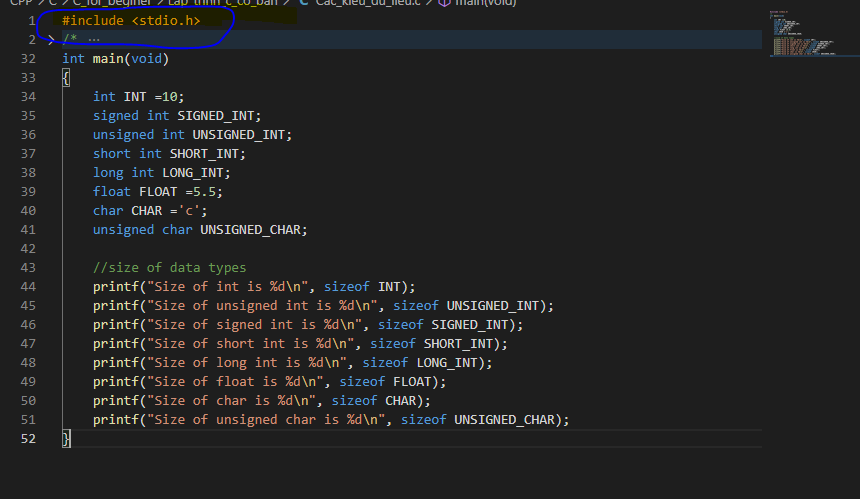
return 0;

}

### 2. Thư viện - Những thứ chuẩn bị sẵn

Thư viện một cách đơn giản hơn là với người lập trình đấy là những file được tạo ra bằng các lệnh, sau đó đóng gói vào một file <\*.c,\*.h>, được khai bao ở phần đầu chương trình bằng lệnh #include <file>.

Ví dụ: **stdio.h** hay **một file nào đấy được lưu ở dạng \*.c hoặc \*.h**

*Hình 3.6 Chương trình c*

Trong một số trường hợp, thì lập trình viên có thể soạn thảo cho mình một thư viện như stack.h , list.h, vector.h để tạo ra một cấu trúc chương trình thuận tiện cho việc xử lý tình huống của vấn đề. Ở trong phần mở rộng tôi sẽ nối xâu hơn về phần này hơn, hướng dẫn các bạn giải quyết tình huống cũng như các vấn đề, bài toán của lập trình, cũng như các bước cơ bản lập trình hoặc tối ưu chương trình của bản thân nhiều hơn.

## IV. HÀM MAIN - HÀM CHÍNH TRONG C

Chúng ta thấy một điều, chương trình máy tính rất giống như ông giám đốc điều hành công ty: xử lý công việc, nhân sự, hợp đồng nội bộ, hợp đồng đối tác,.. Hay là một vị tướng quân trên sa trường đang điều binh khiển tướng. Vậy thì lập trình máy tính và cuộc sống có gì khác và giống nhau, thực ra lập trình máy tính chính là phản ánh cuộc sống, từ nhu cầu của cuộc sống mà được tạo ra.  
Hàm main() trong lập trình là nơi mà khi biên dịch ra chương trình thì trình biên dịch chạy vào hàm main để thực thi toàn bộ lệnh có trong hàm. Các lệnh ngoài hàm sẽ không được thực thi nếu hàm main không gọi tới nó. Một chương trình máy tính luôn chỉ có một hàm main - “Một núi không thể có hai hổ”.

Những bí mật của hàm main:

- Hàm main giữ vị trí độc tôn trong chương trình, tức mỗi chương trình chỉ có 1 main ().

- Hàm main cũng là hàm cũng mang một số đặc tính của hàm nên sẽ có giá trị trả về.

- Tuy nhiên hàm main chỉ có giá trị trả về là int hoặc void.

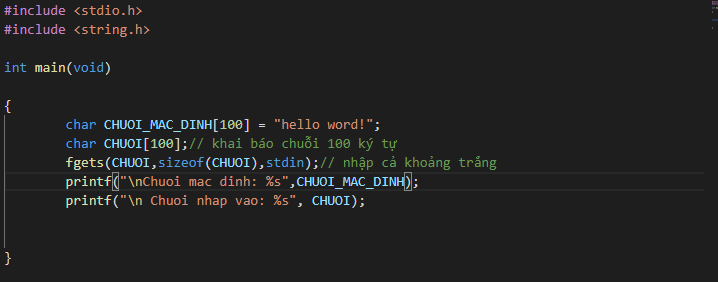
- Những code hay hàm không được gọi tới trong main thì không được thự thi, dù đôi lúc complier cũng bắt lỗi của những đoạn code ấy.

- Biến được khai báo trong main cũng có thể coi là biến bán-toàn cục (giá trị của biến sẽ có hiệu lực từ nó đến cuối chương trình).

**Thông tin thêm***:* *Trước khi người ta tách đoạn lệnh ra khỏi main thành hàm riêng thì toàn bộ code được code tập trung ở tất cả ở main.*

- Ở hàm main sẽ có return 0 ở cuối hàm, nhưng có nó chương trình vẫn chạy,không có vẫn chạy, hoặc trả về kiểu void. Thực ra return 0 được xem như một kiểu tra về giúp cho việc kiểm soát đoạn chương trình viết ra đã xảy ra lỗi hay thực thi toàn bộ. Nếu chúng ta để ý khi sử dụng khai báo là return 0, nếu có lỗi logic xảy ra, ở cuối dòng console sẽ return 1 con số khác 0.

*Thông tin thêm: Khai báo kiểu int main(void) {} sẽ giúp chương trình thực thi nhanh hơn các kiểu khai báo khác.*

*Hình 3.6 Hàm main*

## V. BIẾN VÀ HÀM KHÁC

### 1. Biến

Một **biến trong C** là tên của vị trí bộ nhớ. Nó được sử dụng để lưu trữ dữ liệu. Giá trị của nó có thể được thay đổi và nó có thể được sử dụng lại nhiều lần. Mỗi biến trong C có một loại dữ liệu cụ thể, xác định kích thước của bộ nhớ của biến; phạm vi các giá trị có thể được lưu trữ trong bộ nhớ đó.

*Biến là một cách để thể hiện vị trí bộ nhớ thông qua một cái tên để nó có thể được xác định một cách dễ dàng. Tên của có một biến thể bao gồm các chữ cái, chữ số và ký tự gạch dưới. Nó phải bắt đầu bằng một chữ cái thư hoặc một gạch dưới. Biến trong C có****phân biệt chữ hoa và chữ thường****.*

***Cú pháp khai báo*:**

**<Kiểu dữ liệu> <Tên biến> | <type> <variable\_list>**

Quy tắc khai báo biến trong C

* *Một biến có thể có các chữ cái, chữ số và dấu gạch dưới.*
* *Tên biến chỉ có thể bắt đầu bằng bảng chữ cái và dấu gạch dưới. Nó không thể bắt đầu bằng chữ số.*
* *Không có khoảng trắng trong tên biến.*
* *Tên biến không phải là bất kỳ từ hoặc từ khóa dành riêng như int, float, vv*

.**Tên biến hợp lệ:**

|  |  |
| --- | --- |
| 1  2  3 | . int a;  . int \_ab;  . int a30; |

**Tên biến không hợp lệ:**

|  |  |
| --- | --- |
| 1  2  3 | . int 2;  . int a b;  . int long; |

**Các kiểu biến trong C**

* Biến local (địa phương).
* Biến global (toàn cầu).
* Biến static.
* Biến automatic.
* Biến external.

Lúc chúng ta nhìn vào 5 loại biến, thì việc đầu tiên không nên học thuộc lòng mà hãy nghĩ: “Dựa vào đâu mà phân ra 5 loại biến đó? Sau này có thêm nữa không?”

*Dựa vào tính chất của biến người ta chia làm 2 loại: Tự động chuyển hay cố định, dựa vào địa bàn hoạt động người ta lại chia ra thêm 4 loại nữa.*

1. Biến local (địa phương hay biến cục bộ).

Có nhiều kiểu gọi biến như từ địa phương để dịch cho từ local, hoặc có thể gọi cách khác là cục bộ, ý muốn chỉ ra tác dụng hay ảnh hưởng của biến này chỉ nằm ở 1 hàm nào đấy.

Tham khảo ví dụ sau:

|  |
| --- |
| void function\_1() {      int x = 10; // local variables of function\_2  x +=10;  }  void function\_2() {      int x; // local variables of function\_2  printf(“%d”, x);  } |

Trong ví dụ trên thì rõ ràng trong hai hàm thì điều có biến x, nhưng biến x ở hàm 1 và hàm 2 không liên quan gì đến nhau, dù x ở hàm 1 thay đổi thì không ảnh hưởng đến hàm 2.

** Riêng biến trong hàm main đặc biệt hơn các biến trong hàm khác**

1. Biến global (toàn cầu – toàn cục).

*Một biến được khai báo bên ngoài hàm hoặc khối lệnh được gọi là biến toàn cầu. Bất kỳ hàm nào cũng có thể thay đổi giá trị của biến toàn cầu. Nó có sẵn cho tất cả các chức năng. – Nguồn: Google.com.vn*

Riêng tôi có cách hiểu đơn giản hơn, biến toàn cục là biến có giá trị ảnh hưởng tới toàn bộ hoặc bán toàn bộ chương trình. Giá trị của biến đó có thể ảnh hướng 1 lượt tới tất cả các hàm ( kể cả hàm con và hàm main).

Tham khảo ví dụ sau:

|  |
| --- |
| int a = 20; // global variables  void function1() {      int x = 10; // local variables  printf(“%d”,a);// output : print 20 in console  }  a = a+10;  printf(“%d”,x)// error complier  printf(“%d”,a);// output : print 30 in console |

Trong ví dụ trên

> Biến trong hàm main cũng có thể coi là biến toàn cục.

1. Biến static ( Biến tĩnh )

Biến tĩnh là gì? Biến tĩnh là biến không động. Tôi đùa thôi, nhưng thực sự là vậy, biến tĩnh là biến có giá trị không thay đổi trong quá trình thực thi chương trình. Nó không được phép thay đổi giá trị kể từ khi được khai báo vào gán giá trị.

- Khác với các biến có thể thay đổi giá trị, biến tĩnh phải được gán giá trị từ lúc vừa được khai báo.

- Mọi thao tác làm thay đổi giá trị của biến tĩnh điều bị vô hiệu,ở những chuẩn biên dịch gần đây ( g++ version mới gần nhất) thì việc gán giá trị cho biến tĩnh được compiler báo lỗi trực tiếp.

Tham khảo ví dụ sau:

|  |
| --- |
| **static** int STATIC\_VARIABLES = 100;// static variables |

Trong ví dụ trên, biến STATIC\_VARIABLES được khai báo là biến tĩnh với giá trị 100 không đổi. Biến tĩnh còn gọi là hằng.

1. Biến external (Biến mở rộng).

Trong lập trình c hiện nay, ít thấy nhất là biến dạng external, nó cũng như tên của mình. Nó cho phép các file.c được chia sẽ dữ liệu qua lại với nhau qua biến external. Hiện nay thì người ta không còn sử dụng đến nó nhiều. **Biến này còn có tên gọi khác là biến được cho vào quên lãng.**

Bạn có thể lên google search về biến này để biết thêm.

1. Biến automatic (Biến tự động)

Biến tự động, chức năng cũng như tên biến, tức là biến có thể tự động xác định kiểu dữ liệu của bản thân, khi nhận được giá trị của biến. Nó có thể tùy ý biến thành kiểu char, int hay foat khi nhận được giá trị của kiểu dữ liệu.

Tham khảo ví dụ sau:

|  |
| --- |
| auto AUTO\_VAR =10;      printf("\n%d",AUTO\_VAR ); // out put 10      AUTO\_VAR ='b';      printf("\n%c",AUTO\_VAR );// out put b |

Trong ví dụ trên, trên cùng 1 biến ta có thể tùy ý thay đổi giá trị mặc dù khác kiểu dữ liệu.

#### TÓM TẮT

Vậy trong phần biến, ta cần thấu hiểu được những thứ quan trọng:  
- Biến là tên gọi để người lậ trình và máy tính có thể phân biệt được những dữ liệu khác nhau có cùng kiểu.

- Biến sinh ra để lưu trữ dữ liệu và phân biệt nó với những dữ liệu khác.

- Cách sử dụng biến - khai báo: <Tên kiểu dữ liệu> <Tên biến>

- Sử dụng thông minh cần nắm được ưu nhược điểm của biến

### 2.Kiểu dữ liệu

### 3. Hàm

Hàm là một tập các câu lệnh hoặc hàm khác để xử lý một nhiệm vụ cụ thể.

Về ý nghĩa: hàm dường như là một chương trình con để thực thi một nhiệm vụ nào đó hoặc được tách ra từ khối lệnh cho hàm main hoặc hàm được tách ra đỡ phước tạp.

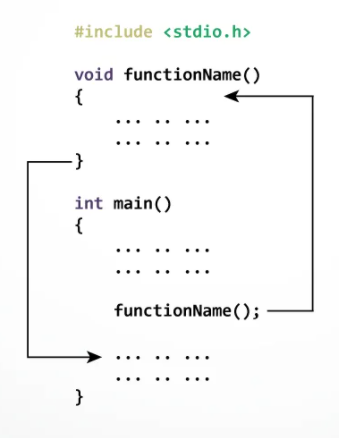
Hàm được xây dựng ở một nơi trong thư viện hoặc chương trình, nhưng được gọi tên **trực tiếp** hay **gián tếp** bởi hàm main.

#### Ưu điểm khi dùng chương trình con:

* + Sau đây là một số ưu điểm nổi bật của sử dụng chương trình con (hàm) mà mình có thể liệt kê, nhưng có 1 điều chắc chắn rằng: Hãy cố gắng thực hành viết code của bạn sử dụng hàm nếu có thể nhé.
  + Sử dụng chương trình con khiến code của bạn trông sáng sủa hơn và gọn gàng, người đọc code sẽ dễ hiểu hơn bằng cách nhìn vào từng hàm con ta có thể dễ dàng xác định vai trò của nó trong chương trình.
  + Dễ dàng quản lý, nâng cấp và tìm lỗi chương trình. Bởi vì bạn biết rõ hàm nào đang làm gì, nếu mà chẳng may gặp lỗi thì bạn cũng nhanh chóng xác định lỗi đó của hàm nào thay vì phải dò từng dòng trong hàm main
  + Viết 1 lần và gọi được ở nhiều nơi: Khi bạn dùng hàm thì bạn chỉ phải viết một lần và gọi tới nó bất cứ khi nào bạn muốn. Bạn cũng có thể đóng gói các hàm đó để sử dụng cho các chương trình khác.

#### Nguyên lý hoạt động của hàm:

Hàm chỉ được thực thi khi được gọi trong hàm main, nhưng lại được complier kiểm lỗi trong cùng một lúc với các hàm khác và hàm main.



*Hình 3.6 Nguyên lý hoạt động của hàm*

Các dạng trả về của hàm:

- Hàm có giá trị trả về.

- Hàm không có giá trị trả về.

Nhược điểm của hàm

## VI. GIỚI THIỆU MỘT SỐ GIẢI THUẬT THÔNG DỤNG

## VII. MỘT SỐ CÁCH PHÂN TÍCH YÊU CẦU ĐỂ GIẢI QUYẾT VẤN ĐỀ

# B. PHẦN MỞ RỘNG

## I. DANH SÁCH

### 1. Danh sách

### 2. Danh sách liên kết

##### a. Danh sách liên kết đơn.

##### b. Danh sách liên kết đôi.

## II. BẢNG BĂM

### 1. Băm trong.

### 2. Băm ngoài

## III. CẤU TRÚC DỮ LIỆU

### 1. Cấu trúc thường.

### 2. Một số dạng cấu trúc khác.

## IV. GIỚI THIỆU MỘT SỐ TỪ KHÓA TRONG C

## V. COMMENTS

## VI. LỖI TRONG C

# VII TÓM TẮT

MỤC LỤC

[LẬP TRÌNH C CƠ BẢN 1](#_Toc63693945)

[A. PHẦN CƠ BẢN 1](#_Toc63693946)

[I. TỔNG QUAN VỀ CHƯƠNG TRÌNH C 1](#_Toc63693947)

[1. Giới thiệu chương trình cơ bản. 1](#_Toc63693948)

[2. Lập trình máy tính căn bản là gì? 2](#_Toc63693949)

[3. Lập trình c là gì? 2](#_Toc63693950)

[TỔNG KẾT 6](#_Toc63693951)

[II. CẤU TRÚC CỦA MỘT CHƯƠNG TRÌNH C 7](#_Toc63693952)

[1. Tiền xử lý. 7](#_Toc63693953)

[Dưới đây là danh sách các thẻ tiền xử lý quan trọng trong C: 7](#_Toc63693954)

[2. Hàm main. 8](#_Toc63693955)

[3. Biến và hàm khác. 9](#_Toc63693956)

[4. Comments trong C. 9](#_Toc63693957)

[TỔNG KẾT 10](#_Toc63693958)

[III. PHẦN HEADER – TIỀN XỬ LÝ 11](#_Toc63693959)

[1. Lệnh tiền xử lý (Directive)- Những phần được Ông đạo cụ sắp xếp 11](#_Toc63693960)

[Hình 3.5 Macro được định nghĩa trước trong C 13](#_Toc63693961)

[2. Thư viện - Những thứ chuẩn bị sẵn 14](#_Toc63693962)

[IV. HÀM MAIN - HÀM CHÍNH TRONG C 15](#_Toc63693963)

[V. BIẾN VÀ HÀM KHÁC 16](#_Toc63693964)

[1. Biến 16](#_Toc63693965)

[2.Kiểu dữ liệu 17](#_Toc63693966)

[3. Hàm 17](#_Toc63693967)

[VI. GIỚI THIỆU MỘT SỐ GIẢI THUẬT THÔNG DỤNG 19](#_Toc63693968)

[VII. MỘT SỐ CÁCH PHÂN TÍCH YÊU CẦU ĐỂ GIẢI QUYẾT VẤN ĐỀ 19](#_Toc63693969)

[B. PHẦN MỞ RỘNG 19](#_Toc63693970)

[I. DANH SÁCH 19](#_Toc63693971)

[1. Danh sách 19](#_Toc63693972)

[2. Danh sách liên kết 19](#_Toc63693973)

[II. BẢNG BĂM 19](#_Toc63693974)

[1. Băm trong. 19](#_Toc63693975)

[2. Băm ngoài 19](#_Toc63693976)

[III. CẤU TRÚC DỮ LIỆU 19](#_Toc63693977)

[1. Cấu trúc thường. 19](#_Toc63693978)

[2. Một số dạng cấu trúc khác. 19](#_Toc63693979)

[IV. GIỚI THIỆU MỘT SỐ TỪ KHÓA TRONG C 19](#_Toc63693980)

[V. COMMENTS 19](#_Toc63693981)

[VI. LỖI TRONG C 19](#_Toc63693982)

[VII TÓM TẮT 20](#_Toc63693983)