# BẮT ĐẦU VỚI NGÔN NGỮ C

## PHẦN 1.1 HELLO WORLD

Để tạo một chương trình C đơn giản in dòng chữ "Hello, World " lên màn hình, sử dụng trình soạn thảo văn bản để tạo file mới (ví dụ:

hello.c — phần mở rộng file phải là .c) chứa mã nguồn sau:

**hello.c**

|  |
| --- |
| # include<stdio.h>  int main(void)  {      puts("Hello, World");      return 0;  } |

**Hãy xem từng dòng trong chương trình đơn giản này**

|  |
| --- |
| # include<stdio.h> |

Dòng mã này yêu cầu trình biên dịch cần thêm nội dung của tệp tiêu đề thư viện chuẩn **stdio.h** vào chương trình.

Tệp tiêu đề thường chứa các khai báo của các hàm, macros và kiểu dữ liệu trong thư viện chuẩn.

và bạn cần phải thêm vào tệp tiêu đề trước khi sử dụng chúng. Dòng này thêm vào thư viện **stdio.h** để nó có thể gọi hàm **puts().**

Tìm hiểu thêm về tệp tiêu đề

|  |
| --- |
| int main(void) |

Dòng mã này bắt đầu định nghĩa của một hàm. Nó cho biết tên của hàm (main), kiểu dữ liệu và số lượng các đối số mà nó mong đợi (void, nghĩa là không có) và loại giá trị mà hàm này trả về (int). Theo nguyên tắc, **main** là một tên đặc biệt, nó chỉ dùng để đặt cho function chính của chương trình, và lúc nào chương trình cũng sẽ bắt đầu từ **function main**.

Việc thực thi chương trình bắt đầu trong hàm main().

|  |
| --- |
| {      ...  } |

Dấu ngoặc nhọn (curly braces) được sử dụng thành cặp để chỉ định vị trí bắt đầu và kết thúc một khối mã(khối lệnh). Chúng có thể được sử dụng theo nhiều cách khác nhau, nhưng trong trường hợp này, chúng chỉ định vị trí bắt đầu và kết thúc của một hàm.

|  |
| --- |
| puts("Hello, World"); |

Dòng này gọi hàm puts() để đưa ra văn bản ra đầu ra tiêu chuẩn (mặc định là màn hình), tiếp theo là một dòng mới.

Chuỗi cần đưa ra được bao gồm trong cặp dấu ngoặc đơn().

"Hello, World" là chuỗi sẽ được ghi ra màn hình. Trong ngôn ngữ C, mọi giá trị chuỗi phải nằm trong dấu ngoặc kép "...".

Tìm hiểu thêm về chuỗi.

Trong các chương trình C, mỗi câu lệnh cần được kết thúc bằng dấu chấm phẩy (tức là **;**).

|  |
| --- |
| return 0; |

Khi chúng ta định nghĩa hàm **main(),** chúng ta khai báo nó là một hàm trả về kiểu **int**, có nghĩa là nó cần trả về một giá trị số nguyên. Trong ví dụ này, chúng ta đang trả về giá trị nguyên là **0**, được sử dụng để chỉ rằng chương trình đã kết thúc thành công. Sau câu lệnh **return 0;**, quá trình thực thi sẽ kết thúc.

**Chỉnh sửa chương trình**

Các trình soạn thảo, chỉnh sửa văn bản đơn giản bao gồm **vim** hoặc **gedit** trên **Linux,** hoặc **Notepad** trên **Windows.** Các trình soạn thảo đa nền tảng khác bao gồm **Visual Studio Code** hoặc **Sublime Text.**

Trình chỉnh sửa phải tạo các tệp văn bản thuần túy, không phải RTF hoặc bất kỳ định dạng nào khác.

**Biên dịch và chạy chương trình**

Để chạy chương trình, file nguồn này (hello.c) trước tiên cần được biên dịch thành file thực thi (ví dụ: **hello** trên hệ thống **Unix/Linux** hoặc **hello.exe** trên **Windows**). Điều này được thực hiện bằng cách sử dụng trình biên dịch cho ngôn ngữ C.

**Biên dịch bằng GCC**

GCC **(GNU Compiler Collection)** là một trình biên dịch C được sử dụng rộng rãi. Để sử dụng nó, hãy mở một thiết bị đầu cuối, sử dụng dòng lệnh để điều hướng đến vị trí của tệp nguồn rồi chạy:

|  |
| --- |
| gcc hello.c -o hello |

Nếu không tìm thấy lỗi trong mã nguồn (hello.c), trình biên dịch sẽ tạo một tệp nhị phân, tên của tệp được cung cấp bởi đối số cho tùy chọn dòng lệnh -o (hello). Đây là tập tin thực thi cuối cùng.

Chúng tôi cũng có thể sử dụng các tùy chọn cảnh báo *-Wall -Wextra -Werror*, giúp xác định các sự cố có thể khiến chương trình bị lỗi hoặc tạo ra kết quả không mong muốn. Chúng không cần thiết cho chương trình đơn giản này nhưng đây là cách thêm chúng:

|  |
| --- |
| gcc -Wall -Wextra -Werror -o hello hello.c |

**Sử dụng trình biên dịch clang**

Để biên dịch chương trình bằng **clang**, bạn có thể sử dụng:

|  |
| --- |
| clang -Wall -Wextra -Werror -o hello hello.c |

Theo thiết kế, các tùy chọn dòng lệnh **clang** tương tự như của **GCC.**

**Sử dụng trình biên dịch Microsoft C từ dòng lệnh**

Nếu sử dụng trình biên dịch Microsoft cl.exe trên hệ thống Windows hỗ trợ Visual Studio và nếu tất cả các biến môi trường được đặt, ví dụ C này có thể được biên dịch bằng lệnh sau. Lệnh này sẽ tạo ra tệp hello.exe có thể thực thi được trong thư mục mà lệnh được thực thi trong (Có các tùy chọn cảnh báo như /W3 cho cl, đại khái là tương tự như -Wall vv cho GCC hoặc clang).

|  |
| --- |
| cl hello.c |

**Thực hiện chương trình**

Sau khi được biên dịch, tệp nhị phân sau đó có thể được thực thi bằng cách nhập ./hello trong thiết bị đầu cuối. Khi thực thi, chương trình đã biên dịch sẽ in **Hello, World**, theo sau là một dòng mới, tới dấu nhắc lệnh.

## PHẦN 1.2 BẢN GỐC “HELLO WORLD!” TRONG K&R C

Sau đây là bản gốc "Hello World!" chương trình từ cuốn sách **The C Programming Language** của BrianKernighan và Dennis Ritchie (Ritchie là nhà phát triển ban đầu của ngôn ngữ lập trình C tại Bell Labs), được gọi là "K&R":

|  |
| --- |
| Version = K&R  #include <stdio.h>  main()  {   printf("hello, world\n");  } |

Lưu ý rằng **The C Programming Language** không được chuẩn hóa vào thời điểm viết ấn bản đầu tiên của cuốn sách này (1978), và chương trình này có thể sẽ không được biên dịch trên hầu hết các trình biên dịch hiện đại trừ khi chúng được hướng dẫn chấp nhận mã C90.

Câu ví dụ đầu tiên trong cuốn sách K&R (**The C Programming Language**) hiện được coi là chất lượng kém, một phần là do thiếu một kiểu trả về rõ ràng cho **main()** và một phần là do thiếu câu lệnh **return**. Phiên bản thứ hai của cuốn sách được viết cho tiêu chuẩn C89 cũ. Trong C89, kiểu của main sẽ mặc định là int, nhưng ví dụ K&R không trả về một giá trị xác định cho môi trường. Trong các tiêu chuẩn C99 và sau này, kiểu trả về là bắt buộc, nhưng có thể bỏ qua câu lệnh **return** của main (**và chỉ main**), vì có một trường hợp đặc biệt được giới thiệu với C99 5.1.2.2.3 - nó tương đương với việc trả về 0, biểu thị thành công.

Dạng chính được khuyến nghị và di động nhất cho các hệ thống được lưu trữ là **int main(void)** khi chương trình không sử dụng bất kỳ đối số dòng lệnh nào, hoặc **int main(int argc, char \*\*argv)** khi chương trình sử dụng các đối số ở các dòng lệnh.

**C90 §5.1.2.2.3 Kết thúc chương trình**

Một câu lệnh **return** từ lời gọi ban đầu tới hàm **main** tương đương với việc gọi hàm **exit** với giá trị trả về của hàm **main** là **đối số**. Nếu hàm main thực hiện một câu lệnh **return** mà không xác định giá trị, trạng thái kết thúc trả về cho môi trường chủ không được xác định.

**C90 §6.6.6.4 Câu lệnh return**

Nếu một câu lệnh **return** mà không có biểu thức được thực thi và giá trị của lời gọi hàm được sử dụng bởi người gọi, hành vi đó là không xác định. Đạt đến dấu **}** kết thúc một hàm tương đương với việc thực thi một câu lệnh **return** mà không có biểu thức.

**C99 §5.1.2.2.3 Kết thúc chương trình**

Nếu kiểu trả về của hàm **main** là một kiểu tương thích với **int**, một câu lệnh **return** từ lời gọi ban đầu tới hàm **main** tương đương với việc gọi hàm **exit** với giá trị trả về của hàm main là đối số; đạt đến dấu **}** kết thúc hàm main sẽ trả về giá trị 0. Nếu kiểu trả về không tương thích với int, trạng thái kết thúc trả về cho môi trường chủ không được xác định.

# COMMENTS

Chú thích (comments) được sử dụng để chỉ điều gì đó cho người đọc mã nguồn. Chú thích được coi như một phần trống trên mã nguồn và không thay đổi bất kỳ ý nghĩa thực sự nào của mã. Có hai cú pháp được sử dụng cho chú thích trong ngôn ngữ C, đó là **/\* \*/** (bản gốc) và **//** (bản mới hơn 1 chút). Một số hệ thống tài liệu sử dụng chú thích được định dạng đặc biệt để hỗ trợ tạo tài liệu cho mã nguồn.

## PHẦN 2.1 CHÚ THÍCH BẰNG CÁCH SỬ DỤNG TIỀN XỬ LÝ(PREPROCESSOR)

Một phần lớn mã nguồn cũng có thể bị "chú thích" bằng cách sử dụng các chỉ thị tiền xử lý **#if 0** và **#endif.** Điều này hữu ích khi mã chứa các chú thích nhiều dòng mà không thể lồng nhau nếu không có cơ chế chú thích này.

|  |
| --- |
| #if 0 /\* Bắt đầu "chú thích", bất cứ điều gì từ đây trở đi sẽ bị loại bỏ bởi tiền xử lý \*/  /\* Một lượng mã lớn với các chú thích nhiều dòng \*/  int foo()  {  /\* một số câu lệnh \*/  ...  /\* ... một chú thích mô tả câu lệnh if ... \*/  if (someTest) {   /\* some more comments \*/  return 1;  }  return 0;  }  #endif /\* 0 \*/  /\* Mã từ đây trở đi không bị "chú thích" (được bao gồm trong tệp thực thi biên dịch) \*/ |

## PHẦN 2.2 CHÚ THÍCH ĐƯỢC GIỚI HẠN BẰNG /\* \*/

Một chú thích bắt đầu với một dấu gạch chéo kề ngay sau đó là một dấu hoa thị (/\*) và kết thúc ngay khi gặp một dấu hoa thị kề ngay sau đó là một dấu gạch chéo (\*/). Mọi thứ nằm giữa các cặp ký tự này đều là chú thích và được coi là một phần trống (tức là bị bỏ qua) bởi trình biên dịch.

|  |
| --- |
| /\* Đây là một chú thích \*/ |

Chú thích ở trên là chú thích trên một dòng. Chú thích kiểu /\* này có thể trải dài trên nhiều dòng, như sau:

|  |
| --- |
| /\* Đây là một chú thích  nhiều dòng \*/ |

Mặc dù điều này không bắt buộc, quy ước phong cách thông thường với chú thích trên nhiều dòng là đặt khoảng trắng và dấu hoa thị ở các dòng sau đầu tiên và đặt /\* và \*/ trên các dòng mới, sao cho chúng xếp thành hàng:

|  |
| --- |
| /\* Đây là   \*một chú   \* thích   \*nhiều dòng  \*/ |

Những dấu hoa thị thêm không có tác dụng chức năng nào đối với chú thích vì chúng không có dấu gạch chéo liên quan.

Các chú thích kiểu /\* này có thể được sử dụng trên một dòng riêng biệt, ở cuối một dòng mã hoặc ngay cả trong các dòng mã:

|  |
| --- |
| /\* Chú thích này nằm trên một dòng riêng biệt \*/  if(x && y) { /\* Chú thích này nằm ở cuối dòng mã \*/      if ((complexCondition1) /\* Chú thích này nằm trong một dòng mã \*/          && (complexCondition2)) {      /\* Chú thích này nằm trong một câu lệnh if, trên một dòng riêng biệt \*/      }  } |

Chú thích không thể lồng nhau. Điều này xảy ra vì bất kỳ /\* nào tiếp theo sẽ bị bỏ qua (là một phần của chú thích) và đầu tiên \*/ gặp được sẽ được coi là kết thúc chú thích. Chú thích trong ví dụ dưới đây sẽ không hoạt động:

|  |
| --- |
| /\* Chú thích bên ngoài, có nghĩa là điều này bị bỏ qua => /\* chú thích nội bộ cố gắng \*/ <= kết thúc chú thích,  không phải chú thích này => \*/ |

Để chú thích các khối mã chứa chú thích kiểu này, mà nếu không sẽ bị lồng nhau, bạn có thể tham khảo ví dụ về Chú thích bằng cách sử dụng tiền xử lý.

## PHẦN 2.3 CHÚ THÍCH ĐƯỢC GIỚI HẠN BẰNG DẤU //

Phiên bản ≥ C99

C99 đã giới thiệu việc sử dụng chú thích trên một dòng theo kiểu C++. Loại chú thích này bắt đầu bằng hai dấu gạch chéo kề nhau và kéo dài đến cuối dòng:

|  |
| --- |
| // Mỗi dòng trong số này đều là một chú thích trên một dòng  // Chú ý rằng mỗi dòng phải bắt đầu bằng  // hai dấu gạch chéo kép liên tiếp (//) |

Loại chú thích này có thể được sử dụng trên một dòng riêng biệt hoặc ở cuối một dòng mã. Tuy nhiên, do chú thích này kéo dài đến cuối dòng, nên chúng không thể được sử dụng trong một dòng mã.

|  |
| --- |
| // Chú thích này nằm trên một dòng riêng biệt  if (x && y) { // Chú thích này nằm ở cuối dòng  // Chú thích này nằm trong một câu lệnh if, trên một dòng riêng biệt  } |

## PHẦN 2.4 RỦI RO CÓ THỂ XẢY RA DO TRIGRAPH

Trong quá trình viết chú thích được giới hạn bởi //, có thể xảy ra lỗi chính tả ảnh hưởng đến hoạt động dự kiến của chú thích. Nếu bạn gõ sai như sau:

|  |
| --- |
| int a = 10 //Sao tôi làm điều này??/ |

Dấu gạch chéo / ở cuối là một lỗi chính tả nhưng giờ đây sẽ được hiểu là dấu gạch chéo ngược . Điều này xảy ra vì ??/ tạo thành một trigraph. Trigraph ??/ thực tế là một cách viết dài cho dấu gạch chéo ngược , đây là ký hiệu tiếp tục dòng. Điều này có nghĩa là trình biên dịch sẽ hiểu rằng dòng tiếp theo là tiếp tục của dòng hiện tại, tức là tiếp tục của chú thích, điều này có thể không phải ý định ban đầu.

|  |
| --- |
| int foo = 20; // Bắt đầu từ 20 ??/  int bar = 0;  // Dòng sau sẽ gây lỗi biên dịch (biến 'bar' chưa được khai báo)  // vì 'int bar = 0;' là một phần của chú thích trên dòng trước đó  bar += foo; |