

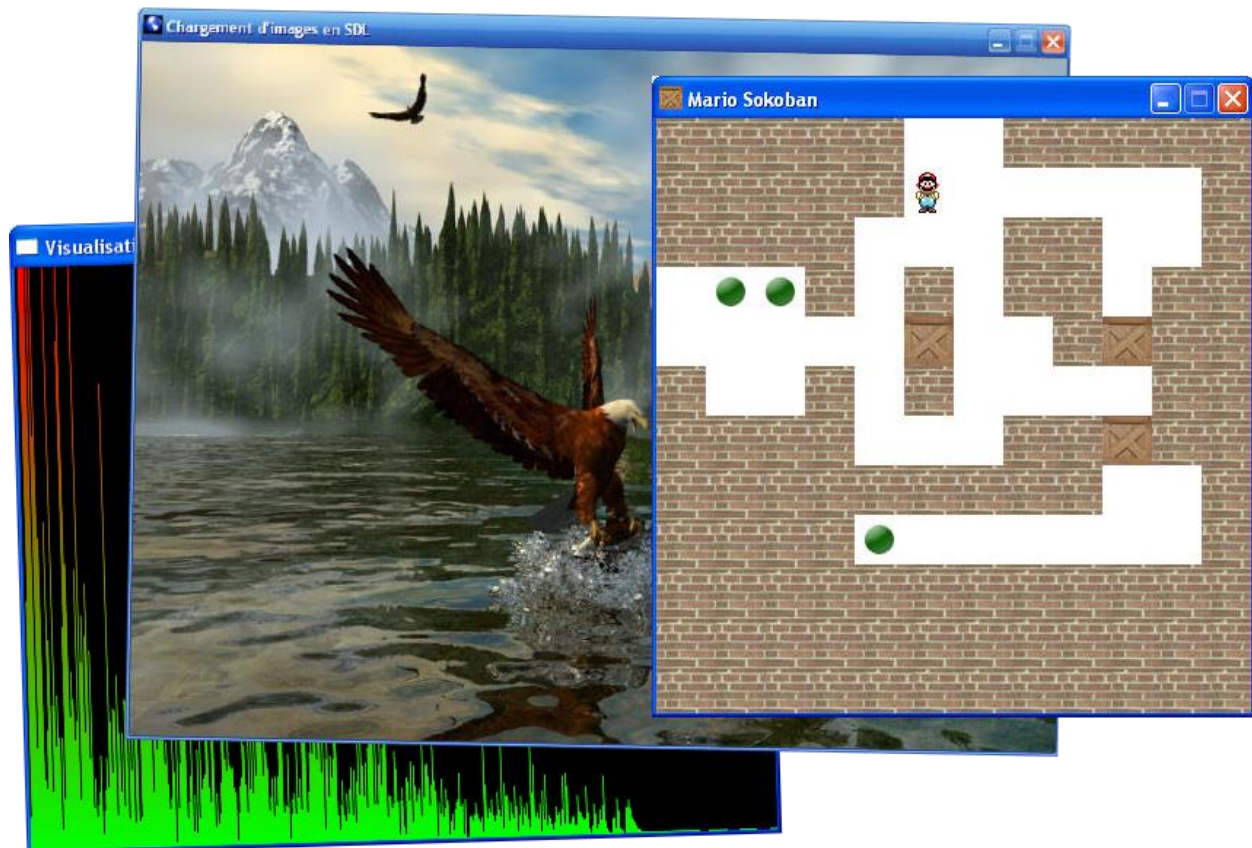
Bài 3: Chương trình đầu tiên của bạn

Chúng ta đã chuẩn bị xong sân chơi, chúng ta sẽ bắt đầu cuộc chơi ngay bây giờ, bạn đang cảm thấy thế nào?

Mục đích của phần hướng dẫn này giúp bạn có thể tạo ra **chương trình đầu tiên** cho chính mình!

Chương trình đầu tiên của bạn:

- Console hay cửa sổ ?
- Đoạn mã tối thiểu
- Viết một tin nhắn lên màn hình
- Những chú thích, khá tiện dụng !
- TRẮC NGHIỆM KIẾN THỨC.



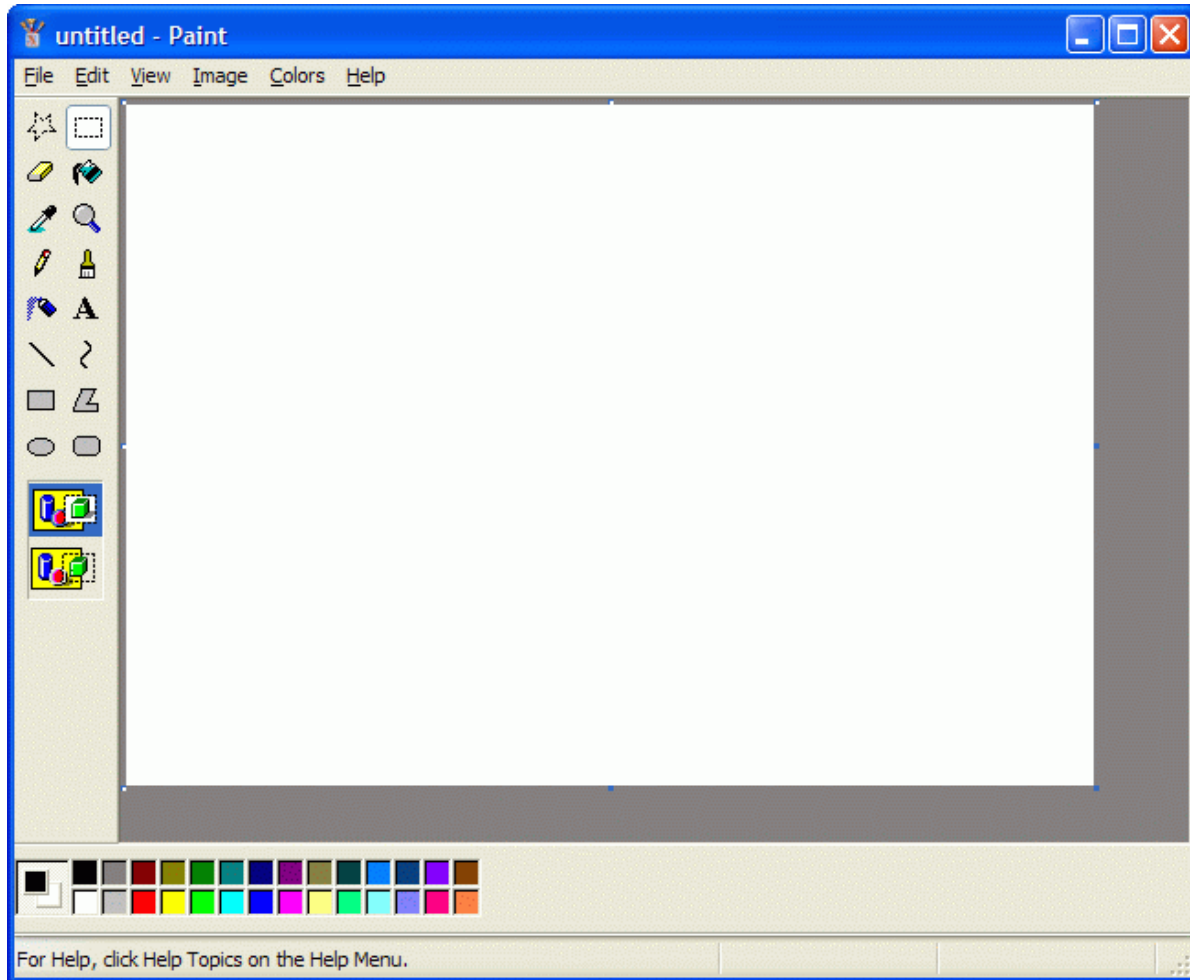
Console hay là cửa sổ?

Có 2 loại chương trình :

- Những chương trình dạng cửa sổ.
- Những chương trình dạng console.

Những chương trình dạng cửa sổ:

Tôi nghĩ rằng các bạn đã biết cái này, lấy một ví dụ điển hình:



Chương trình paint

Đó là một chương trình dạng cửa sổ, các bạn rất muốn tạo ra những chương trình như thế này đúng không?

Với C, chúng ta hoàn toàn có khả năng làm được. Nhưng các bạn chưa đủ sức tạo ra chúng vào lúc này. 😊

Tốt hơn là ta bắt đầu với việc tạo ra một chương trình dạng console.

❓ Nhưng chương trình dạng cửa sổ có giống với những chương trình dạng console không?

Những chương trình dưới dạng console:

Console chính là những chương trình xuất hiện đầu tiên trên thế giới. Vào thời kì đó, máy tính chỉ có khả năng tạo ra những dòng chữ đen và trắng và không đủ mạnh để hiển thị những cửa sổ nhiều màu sắc và hiệu ứng như bạn thấy hiện nay.

Sau đó, Windows đã cho ra đời máy tính có khả năng chạy những chương trình dạng cửa sổ. Vì vậy mà sản phẩm của họ được dùng rộng rãi, khiến phần lớn người sử dụng quên mất sự tồn tại của console.

Và tôi chắc là bạn đang muốn biết console là gì phải không? 😓

Tôi có một tin rất mới cho bạn đây! **console vẫn tồn tại!** Linux đã giữ lại sở thích sử dụng console. Và đây là hình dạng của console trên Linux:

```
2.2.5_appli.html    3.2.7.css          3.6.8.html
2.2.5.css           3.2.8.css          3.6.9.html
2.2.6_appli.html    3.2.9_appli.html   an cres.html
2.2.6.css           3.2.9.css          base.php
2.3.10_appli.html   3.3.10.html        cible_formulaire.php
2.3.10.css          3.3.11.css         cible.html
2.3.11_appli.html   3.3.12.css         design1.css
2.3.11.css          3.3.13_appli.html  erreur_parapgraphe.html
2.3.12.css          3.3.13.css         essai2.css
2.3.13.html         3.3.14_appli.html  essai.css
2.3.14.css          3.3.14.css         images
2.3.15.html         3.3.15.css         tests_design.html
2.3.16.css          3.3.1.css          traitement.php
2.3.17.css          3.3.2.html
2.3.18.css          3.3.3.css
[root@ns1 exemples]# cd ..
[root@ns1 xhtml-css]# ls
anins             css.php            images             pseudoformats.php
annexes           design.php         images.php         qcm.php
autres            exemples           index.php          tableaux.php
boites_partiel.php formatage_partiel.php intro.php          texte.php
boites_partie2.php formatage_partie2.php liens.php          xhtml.php
conclusion.php    formulaires.php    listes.php
[root@ns1 xhtml-css]#
```

Một ví dụ về console trong Linux

Đó là console và những đặc điểm cần chú ý là:

- Console ngày nay không chỉ hiển thị trắng và đen.
- Console không được những người mới sử dụng chào đón lắm.
- Console là một công cụ mạnh mẽ nếu như chúng ta biết cách sử dụng.

Viết một chương trình dạng console đơn giản và lý tưởng hơn cho những người mới học lập trình (sẽ không hề đơn giản nếu bắt đầu học bằng cách tạo ra một chương trình dạng cửa sổ)

Ghi thêm rằng, console ngày nay đã được cải tiến rất nhiều: hiển thị được nhiều màu sắc, và bạn có thể đặt một hình ảnh nào đó lên nền của console. Và đây là hình ảnh một console đã được tạo dựng khá hoành tráng trên HĐH linux 🤖



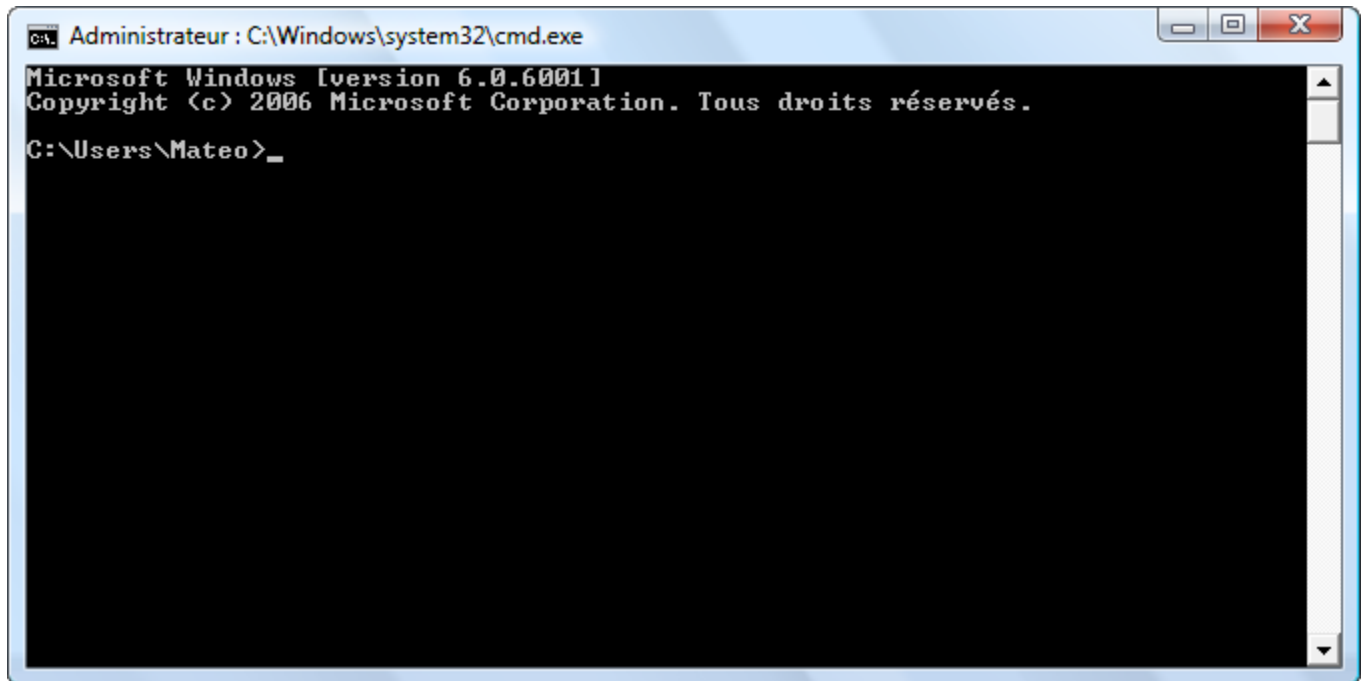
```
[ Wed Aug 17 03:55:04 ]
@ ~ ]--> cd .config
[ Wed Aug 17 03:55:26 ]
@ ~/.config ]--> ls
chromium  geeqie      hotkeys      nitrogen    Thunar      volumeicon
dconf     gnome-mplayer leafpad       openbox     tint2       xfce4
deadbeef  gpview       lxterminal   synapse     transmission xms2
dmenu     gtk-2.0      nautilus     terminator  Trolltech.conf

[ Wed Aug 17 03:55:30 ]
@ ~/.config ]--> cd openbox
[ Wed Aug 17 03:55:34 ]
@ ~/.config/openbox ]--> ls
autostart.sh  menu.xml  pipemenu    rc.xml  rc.xml.bak  scripts
[ Wed Aug 17 03:55:40 ]
@ ~/.config/openbox ]--> sudo leafpad rc.xml
Password:
[ Wed Aug 17 03:58:48 ]
@ ~/.config/openbox ]--> cd
[ Wed Aug 17 03:58:56 ]
@ ~ ]--> cd Images
[ Wed Aug 17 03:58:59 ]
@ ~/Images ]--> scrot -d 3 screenshot.png
Taking shot in 3.. 2.. 1.. 0.
[ Wed Aug 17 03:59:20 ]
@ ~/Images ]--> scrot -cd 3 screenshot.png
Taking shot in 3.. 2.. 1.. 0.
```

Hê hê.. khá kinh dị 🤖

? Trên hệ điều hành Windows có console hay không?

Có nhưng nó đã bị giấu đi, ta có thể nói như thế. 😊
Bạn có thể gọi nó bằng cách vào Start => run => nhập "cmd".
Và đây chính là console của Windows, thật kì diệu:



Console trên Windows

Nếu bạn đang sử dụng Windows, chương trình đầu tiên bạn sắp tạo ra sẽ tương tự như thế.
Với việc bắt đầu từ console, bạn sẽ học được những kiến thức lập trình nền tảng cần thiết để có thể tạo ra những chương trình dạng cửa sổ về sau nên đừng nản chí nhé!

Những dòng code tối thiểu cần phải có.

Trên bất kỳ công cụ lập trình nào, chúng ta đều phải viết ra ít nhất một đoạn code, tuy rằng chúng không thực hiện điều gì nhưng đó là điều bắt buộc.
Đó là đoạn code tối thiểu mà ta sắp sửa tìm hiểu ngay sau đây. Hầu hết các chương trình viết bằng ngôn ngữ C đều phải sử dụng.

Tôi sẽ sử dụng IDE (Integrated Development Environment) **Code::Blocks** để hướng dẫn bạn.
Điều bạn cần làm sau khi mở **Code::Blocks** là tạo một project mới như tôi đã hướng dẫn ở bài trước (vào menu chọn *File / New / Project...*, chọn *Console Application* và chọn **ngôn ngữ C**).

Code::Blocks đã tạo sẵn một đoạn mã tối thiểu mà chúng ta cần:

C code:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf ("Hello world!\n");
    return 0;
}
```



Cần ghi chú là có một dòng trắng ở cuối đoạn code. Được thực hiện bằng cách nhấn phím "ENTER" sau dấu " } ". Mọi tập tin C bình thường đều phải kết thúc bằng một dòng trắng và cũng không có gì nghiêm trọng nếu bạn không thực hiện nó, chỉ là compiler có thể sẽ hiển thị một thông tin warning để thông báo.

Ghi chú thứ 2 là dòng

```
int main ( )
```

cũng có thể được viết thành:

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

Cả hai cách viết đều đúng, nhưng cách viết thứ 2 thông dụng hơn rất nhiều. Tôi sẽ sử dụng cách viết này ở những bài hướng dẫn kế tiếp. Hiện giờ, bạn có sử dụng cách viết nào cũng không quan trọng vì ta vẫn chưa có đủ kiến thức để hiểu được ý nghĩa và cách hoạt động của chúng.

Nếu bạn đang sử dụng một IDE khác, hãy copy đoạn code ở trên vào file main.c

Hãy lưu lại. Tôi biết là chúng ta vẫn chưa làm gì cả, nhưng hãy lưu lại, đây là một thói quen tốt cần tập. Bình thường bạn chỉ dùng duy nhất một file source "main.c" (những file còn lại là file project được tạo bởi IDE của bạn).

Ý nghĩa đoạn mã tối thiểu ở trên:

Đoạn code đó với bạn thật rắc rối nhưng với tôi đó là đoạn code hiển thị một tin nhắn lên màn hình.

Chúng ta bắt đầu học cách đọc và hiểu chúng. 😊

Bắt đầu từ 2 dòng đầu tiên, chúng có vẻ giống nhau:

C code

```
#include <stdio.h>
#include <stdlib.h>
```

Đây chính là những dòng đặc biệt thường thấy ở đầu những file source và dễ dàng nhận biết vì nó bắt đầu từ dấu “#”. Ta gọi chúng là **preprocessor directives (những chỉ thị tiền xử lý)** vì nó sẽ được đọc bằng một chương trình gọi là preprocessor (chương trình tiền xử lý), chương trình này sẽ chạy đầu tiên khi ta thực hiện compilation.

Chúng ta đã thấy hình vẽ đơn giản về compilation ở chương trước. Nhưng quá trình đó thực sự không hề dễ dàng như vậy, có rất nhiều thứ diễn ra trong đó. Tôi sẽ nói sau này, tại thời điểm hiện tại, các bạn chỉ cần biết cách viết những dòng đầu tiên vào file của bạn là đủ.

❓ Nhưng những dòng đó nghĩa là gì? Tôi rất muốn biết điều đó!

Từ “include” tiếng Anh có nghĩa là đặt vào, bao gồm. Nó cho phép thêm vào project một số file. Những file này sẽ được sử dụng trong quá trình compilation.

Ở đây có 2 dòng, vậy là sẽ có 2 file được thêm vào. Những file này có tên là *stdio.h* và *stdlib.h*. Đó là những file đã tồn tại trước đó trong source và luôn sẵn sàng khi bạn gọi ra. Chúng ta thường gọi nó là **thư viện (library)**. Và những file này chứa những đoạn code được viết sẵn cho phép hiển thị một đoạn văn lên màn hình.

⚠ **Ghi chú:** Thư viện tiếng anh là “library”. Bạn hãy nắm vững nghĩa dịch chính xác của nó. Tôi nghĩ việt nam mình chỉ gọi là thư viện thôi nhỉ?

Nếu không có những file thư viện đó, ta không thể nào ghi được một đoạn văn lên màn hình.

Về nguyên tắc, máy tính của bạn sẽ không hiểu gì cả.

Tóm lại, 2 dòng đầu tiên đó cho phép ta ghi một tin nhắn lên màn hình "dễ dàng". 😊

C code:

```
int main ( )  
{  
    printf ("Hello world!\n");  
    return 0;  
}
```

Cái mà bạn thấy ở trên, người ta gọi đó là một **function**. Một chương trình C hầu như cấu tạo bởi các function, Tại thời điểm này, chương trình của chúng ta chỉ có một function duy nhất.

Một function cho phép chúng ta tập hợp lại các lệnh cho máy tính, những lệnh này cho phép ta thực hiện chính xác một điều gì đó. Ví dụ, ta có thể viết một function “mở_một_tập_tin” trong đó chứa đựng những chỉ dẫn về cách mở một tập tin cho máy tính.

Lợi ích là, một khi function đã được viết ra, bạn không cần phải nói thêm gì nữa cả. Máy tính sẽ biết làm việc đó bằng cách nào. 😊

Vẫn còn quá sớm để chúng ta tìm hiểu chi tiết về những thành phần cấu tạo nên một function. Chúng ta chỉ xem xét những phần chính của nó. Ở câu đầu tiên, chữ thứ hai (**main**) là tên của function. 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.

Một function luôn có mở đầu và kết thúc, giới hạn bởi những dấu { và }. Tất cả function main của chúng ta đều nằm trong đó. Nếu bạn đã theo kịp những gì tôi đã nói, thì function main của chúng ta gồm 2 dòng:

C code:

```
printf ("Hello world!\n");  
return 0;
```

Ta gọi những dòng nằm trong một function là các **instruction**. (Hãy nắm vững những từ ngữ này 😊).

(instruction: chỉ thị, chỉ dẫn, câu lệnh)

Mỗi một instruction là một lệnh dành cho máy tính, và nó yêu cầu máy tính phải thực hiện chính xác một hành động gì đó.

Như tôi đã nói với bạn, công việc của những người lập trình là động não để viết những instruction, và khi bạn đã thành thục, bạn sẽ có thể tạo ra những function như function “mở_một_tập_tin” hay function “nhân_vật_đi_tới” trong một game nào đó. 😊

Một chương trình không gì khác hơn là tạo nên một dãy các instruction: instruction “hãy làm cái này” instruction “hãy làm cái kia”... Bạn ra những lệnh đã được sắp đặt và máy tính sẽ thực hiện các lệnh đó.



Quan trọng: Tất cả các instruction đều kết thúc bằng một dấu chấm phẩy “;”. Hay nói khác hơn đó là đặc điểm nhận biết một instruction. Nếu bạn quên chúng, chương trình của bạn sẽ không dịch được.

Dòng đầu tiên:

C code:

```
printf("Hello world!\n");
```

Yêu cầu máy tính hiển thị lên màn hình "Hello world!". Khi chương trình bạn chạy đến dòng này, nó sẽ hiển thị tin nhắn ra màn hình, sau đó chuyển sang instruction kế tiếp.

C code:

```
return 0;
```

Có nghĩa là đã kết thúc, 😊 dòng này biểu thị rằng ta đã đến giai đoạn kết thúc function main và yêu cầu gửi giá trị 0.

❓ Vậy thì tại sao chương trình phải trở về số 0?

Trên thực tế, mỗi chương trình khi kết thúc sẽ gửi về một giá trị, ví dụ như để nói rằng tất cả hoạt động tốt (0 = tất cả hoạt động tốt, những số khác có nghĩa là “error”). Hầu như những giá trị này không hề được sử dụng, nhưng thực tế nó vẫn tồn tại.

Chương trình của bạn cũng có thể chạy khi không có **return 0**; nhưng sẽ chính xác và đúng hơn nếu ta thêm vào. 🤖

Vậy là! Chúng ta đã tìm hiểu một ít về cách hoạt động của đoạn mã tối thiểu trên.

Hẳn là các bạn vẫn còn một số nghi vấn khác vì chúng ta đã không tìm hiểu sâu lắm. Nhưng bạn hãy yên tâm, tất cả những câu hỏi sẽ từng tí từng tí một được giải đáp. Tôi không muốn giải thích cho bạn tất cả trong một lần, nếu không đầu óc bạn sẽ hoàn toàn rối bেম, tôi đảm bảo. 😊

Đến giờ, bạn vẫn theo kịp tôi đúng không? Bạn không cần thiết phải cố gắng đọc hết một mạch đâu. Hãy nghỉ ngơi và sau đó làm việc với tinh thần minh mẫn nhất.

Tất cả những gì tôi vừa hướng dẫn cho bạn đều là nền tảng, còn nếu bạn cảm thấy không có vấn đề gì thì ta tiếp tục.

Tôi sẽ vẽ cho bạn lại một biểu đồ tổng hợp với những từ ngữ ta vừa học:

```
#include <stdio.h>
#include <stdlib.h> } Preprocessor Directives

int main()
{
    printf("Hello world!\n");
    return 0; } Instructions } Function
```

Test chương trình

Nhanh thôi, bạn chỉ cần biên dịch chương trình rồi chạy. (Nhấn vào nút Build & Run trong Code::Blocks).

Nếu bạn vẫn chưa lưu file lại, Code::Blocks sẽ yêu cầu bạn save file lại, hãy thực hiện điều đó.



Nếu compilation không thực hiện được và bạn có lỗi dạng “My-program - Release” uses an invalid compiler. Skipping... Nothing to be done ...” Điều đó có nghĩa là bạn đã tải và sử dụng phiên bản Code::Blocks không có **mingw** (compiler). Hãy quay về site Code::Blocks tải về phiên bản có mingw.

Và đây là chương trình đầu tiên của bạn:

```
C:\Users\Mateo\Projets\test\bin\Release\test.exe
Hello world!
Process returned 0 (0x0)   execution time : 0.021 s
Press any key to continue.
_
```

Chương trình đầu tiên của bạn!

Chương trình hiển thị "Hello world!" (dòng thứ nhất).

Những dòng kế tiếp được tạo ra bởi Code::Blocks và giải thích rằng chương trình đã được chạy trong khoảng thời gian 0.021s kể từ lúc bắt đầu.

Sau đó Code::Blocks yêu cầu bạn nhấn vào một phím bất kì để đóng cửa sổ lại. Chương trình của bạn sẽ dừng lại.

Vâng, tôi biết rằng cái đó chẳng có nghĩa gì cả, giống như một trò đùa nhưng đó là tất cả những gì bạn vừa học được. 😊

Nhưng dù sao, đó cũng là chương trình đầu tiên của bạn, hãy nhớ lại cảm giác đó, có thể nó sẽ theo bạn suốt cả đời đây. 😊

Không phải vậy sao ?...

Trước khi bạn cho tôi thấy vẻ mặt của bạn lúc này, tôi xin phép chúng ta bước sang phần tiếp theo, không chậm trễ. 😊

Viết một tin nhắn lên màn hình

Kể từ bây giờ, chúng ta sẽ tự viết code của mình vào chương trình.

Nhiệm vụ của các bạn là hiển thị tin nhắn “Xin chào” lên màn hình.

Giống như trước đó console sẽ mở ra. Tin nhắn “Xin chào” sẽ xuất hiện trong đó.

❓ **Làm cách nào để viết một tin nhắn lên màn hình?**

Việc này khá đơn giản. Nếu bạn sử dụng lại đoạn code ở trên, bạn chỉ cần thay "Hello world!" bằng "Xin chào" trong câu có chứa printf.

Tôi đã nói printf là một **instruction**. Nó ra lệnh cho máy tính: “*Hãy hiển thị cho tôi một tin nhắn lên màn hình*”.

Cần biết thêm rằng **printf** là một function đã được viết bởi những lập trình viên đi trước.

❓ **function này ở đâu ? Tôi chỉ thấy tồn tại mỗi function main mà thôi !**

Bạn có nhớ hai dòng này chứ ?

C code:

```
#include <stdio.h>
#include <stdlib.h>
```

Tôi đã nói với bạn rằng nó cho phép ta thêm vào chương trình những thư viện. Và những thư viện đó chứa đầy những function đã được viết sẵn bên trong. `stdio.h` chứa đựng những function cho phép hiển thị một cái gì đó lên màn hình (ví dụ như function `printf`), nhưng nó đòi hỏi người sử dụng phải đánh ra một cái gì đó (đây là những function mà ta sẽ thấy sau này).

Máy tính, chào bạn đi!

Trong function `main`, chúng ta gọi function `printf`.

Để gọi một function rất đơn giản: ta chỉ cần ghi ra tên của nó, kế tiếp là mở ngoặc đóng ngoặc "()", và một dấu chấm phẩy ";".

```
printf ( );
```

Nhưng công việc của bạn vẫn chưa xong đâu. Chúng ta phải cho function `printf` một tin nhắn để hiển thị. Hãy mở ngoặc () sau `printf`. Trong đó, mở ngoặc kép " ". Cuối cùng đánh điều gì bạn cần máy tính hiển thị bên trong.

C code:

```
printf ("Xin chào");
```

Tôi hi vọng rằng bạn không quên mất dấu chấm phẩy ";" ở cuối cùng, tôi nhắc lại là nó rất quan trọng! Nó cho phép máy tính hiểu rằng instruction của ta kết thúc ở đây.

Và đây là code source mà bạn phải có được:

C code:

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{
    printf ("Xin chào");
    system ("PAUSE");
    return 0;
}
```

Chúng ta có 3 instruction yêu cầu máy tính thực hiện:

1. Hiển thị “Xin chào” lên màn hình.
2. Đưa chương trình vào giai đoạn nghỉ, hiển thị tin nhắn "*Press any key to continue*" và chờ đợi cho đến khi ta đánh thêm 1 phím bất kì lên bàn phím để chuyển sang instruction tiếp theo.
3. Function *main* kết thúc, trả về 0. Chương trình kết thúc.

❓ Việc đưa chương trình vào trạng thái nghỉ có ý nghĩa như thế nào? Chúng ta có được phép xóa đi câu lệnh `system("PAUSE")` hay không?

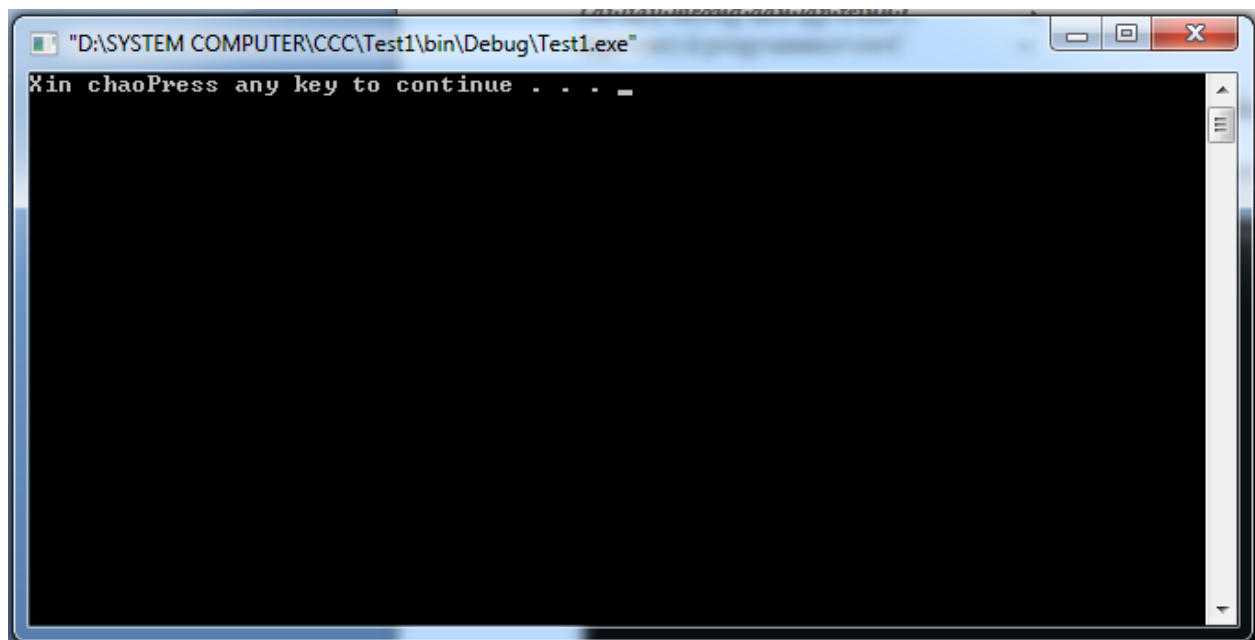
Có chứ, 😊 chắc chắn là bạn có thể. Hãy thử chạy chương trình không có instruction này và bạn sẽ thấy.

Chương trình sẽ không dừng lại. Nói rõ hơn là, máy tính sẽ hiển thị tin nhắn “Xin chào” và tắt chương trình. Cửa sổ của console sẽ hiện ra và biến mất với vận tốc ánh sáng, bạn sẽ không có đủ thời gian để nhận ra điều gì.

Thật ngu ngốc, phải không? 😂

Ghi thêm là, với một số IDE, như là tôi đã nói trước đó, nó sẽ tự động dừng lại ở cuối chương trình. Trong trường hợp đó instruction `system("PAUSE")` coi như vô dụng, bạn có thể xóa nó đi.

Và chúng ta hãy test chương trình với pause, và nó sẽ hiển thị:



Cuối cùng, chương trình hiển thị "Xin chào" đã được hoàn thành.

❓ Nhưng thật sự nó không hoàn toàn hiển thị “xin chào”, có một dòng khác cùng hiển thị sau nó.

Thưa bạn, không có việc gì nghiêm trọng ở đây cả, chúng ta sẽ học cách sửa chữa nó ngay đây.

Bạn muốn kết quả sẽ đưa ra màn hình một dòng khác nằm dưới dòng “Xin chào” của chúng ta, tương tự như việc gõ phím "enter" để xuống dòng khi chat vậy.

Tất nhiên khi chat hay viết code source bạn sẽ xuống dòng bằng cách nhấn enter, nhưng chúng ta đang nói đến việc xuống dòng cho đoạn văn được in ra màn hình console.

Để làm điều đó chúng ta phải sử dụng những kí tự đặc biệt.

Những kí tự đặc biệt:

Những kí tự đặc biệt là những kí tự cho máy tính hiểu rằng ta muốn xuống dòng hay nhấn tab để cách khoảng ...

Những kí tự này tương đối dễ dàng nhận biết. Trước chúng lúc nào cũng có một dấu anti-slash “\”, kế tiếp là một chữ cái hay một số, `\n` và `\t` là 2 kí tự đặc biệt được sử dụng khá thường xuyên mà bạn chắc chắn cần dùng. Bên cạnh đó tôi sẽ cung cấp cho bạn 1 danh sách các ký tự đặc biệt khác để tham khảo trong trường hợp bạn cần đến chúng.

| Escape sequence | Hex value in ASCII | Character represented |
|-------------------|--------------------|---|
| <code>\a</code> | 07 | Alarm (Beep, Bell) |
| <code>\b</code> | 08 | Backspace |
| <code>\f</code> | 0C | Formfeed |
| <code>\n</code> | 0A | Newline (Line Feed); see notes below |
| <code>\r</code> | 0D | Carriage Return |
| <code>\t</code> | 09 | Horizontal Tab |
| <code>\v</code> | 0B | Vertical Tab |
| <code>\\</code> | 5C | Backslash |
| <code>\'</code> | 27 | Single quotation mark |
| <code>\"</code> | 22 | Double quotation mark |
| <code>\?</code> | 3F | Question mark |
| <code>\nnn</code> | any | The character whose numerical value is given by <i>nnn</i> interpreted as an octal number |
| <code>\xhh</code> | any | The character whose numerical value is given by <i>hh</i> interpreted as a hexadecimal number |

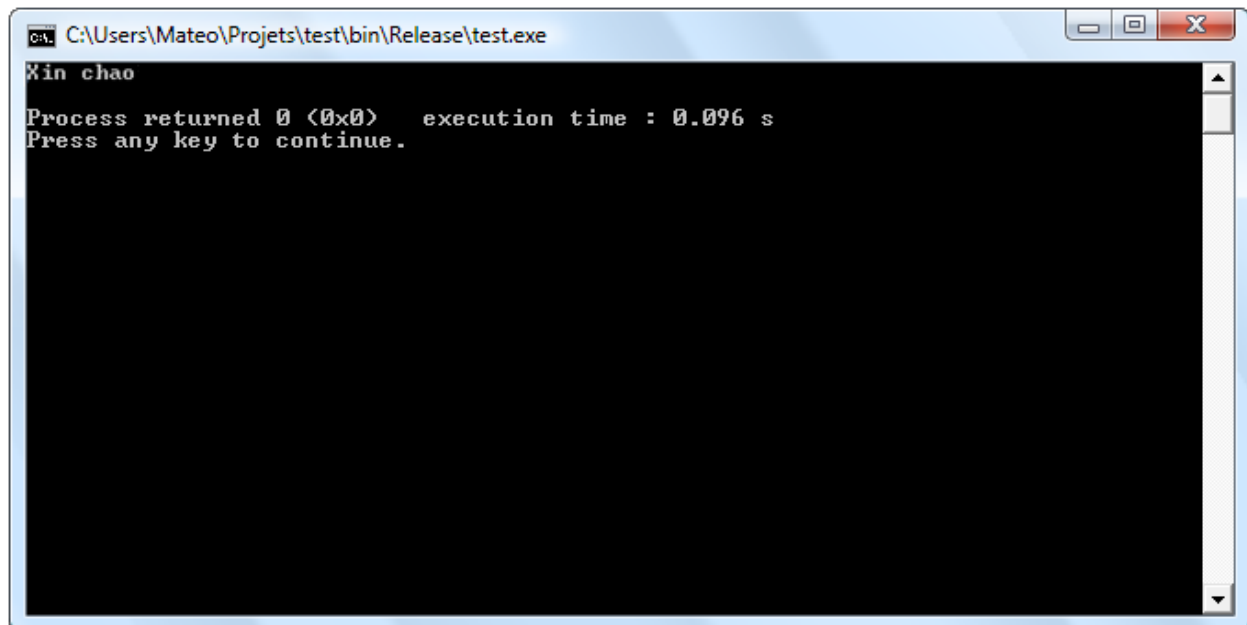
Danh sách các ký tự đặc biệt bạn có thể sử dụng khi lập trình

Trong trường hợp này, chúng ta chỉ cần thêm vào `\n` để xuống dòng.

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{
    printf ("Xin chao"\n);
    system ("PAUSE");

    return 0;
}
```

Và bây giờ chương trình của bạn đã rõ ràng hơn rồi.



Một chương trình hiển thị rõ ràng

⚠️ Bạn có thể viết trong printf duy nhất một ký tự `\n`, điều đó có nghĩa là bạn muốn xuống dòng ở câu kế tiếp. Bạn hãy tập viết những câu thể này:
`printf ("Xin chao\nTam biet\n");`
Nó sẽ hiển thị “Xin chao” ở câu đầu tiên và “Tam biet” ở câu kế tiếp.

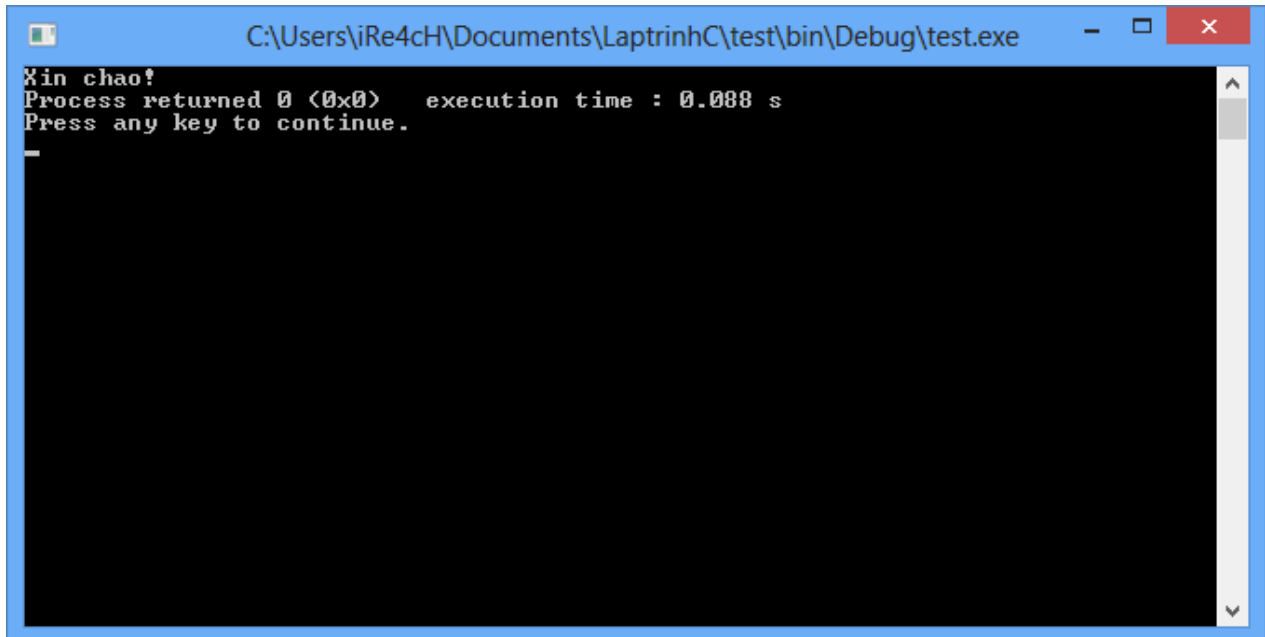
Ví dụ khi sử dụng Code::Blocks phiên bản mới:

Cũng là một chương trình in ra màn hình câu “Xin chào!” nhưng khi viết bằng Code::Blocks phiên bản hiện tại thì chỉ đơn giản như sau:

C code:

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
    printf ("Xin chào!");
    return 0;
}
```

Chương trình hiển thị:



Code::Blocks phiên bản mới chương trình tự động dừng lại và tự động xuống dòng ở cuối.

Bạn có thể nhận ra rằng khi dùng Code::Blocks, bạn không cần phải thêm `\n` để xuống dòng cũng như câu lệnh `system ("PAUSE")` để dừng chương trình như những dòng code tôi đã hướng dẫn bạn trước đó.

Code::Blocks đã thay chúng ta làm việc đó (IDE này khá thông minh đúng không) 😊

Hội chứng Gérard

❓ Xin chào, tôi tên là Gérard và tôi muốn sửa đổi chương trình với tên là “Hello Gérard”. Chỉ vậy thôi, nhưng thật bất ngờ khi máy tính không hiển thị chính xác những gì tôi muốn. Tôi phải làm gì bây giờ?

Đầu tiên xin chào bạn, Gérard 😊

Đây là một câu hỏi khá hay dành cho tôi, và tôi rất vui khi thấy rằng bạn đã bắt đầu có những ý tưởng cải tiến chương trình.

Và đây là câu trả lời, tôi có một tin hơi buồn dành cho bạn: console trên Windows không hiển thị được những dấu trọng âm, nhưng ngược lại trên Linux ta có thể làm điều đó. 😊

Trong trường hợp này bạn có 2 lựa chọn:

- **Chuyển sang Linux:** lựa chọn này khá là phức tạp vì lúc đó tôi phải giải thích cho riêng bạn cách sử dụng Linux. Nếu bạn chưa đủ trình độ để sử dụng vào lúc này, hãy quên lựa chọn này đi.
- **Không sử dụng những dấu trọng âm.** Cách này hơi miễn cưỡng nhưng lúc này bạn phải lựa chọn nó. Console của Windows có những hạn chế, nó chỉ hiển thị những tin nhắn không có dấu.

Và bạn sẽ ghi là:

C code:

```
printf("Hello Gerard\n");
```

Tôi xin cảm ơn bạn Gérard đã giúp tôi nhớ lại vấn đề này

ps: Nếu tên các bạn cũng có dấu như bạn Gérard, thì cũng làm tương tự vậy nhé.

Những lời chú thích, vô cùng tiện dụng!

Trước khi kết thúc phần này, tôi nhất thiết phải chỉ cho bạn một cái khá hay, mà ta gọi chúng là các comment. Trên các ngôn ngữ lập trình ta luôn có thể thêm vào những ghi chú vào trong mã nguồn của bạn. Và đối với ngôn ngữ C bạn cũng có thể làm như vậy.

Có nghĩa là bạn thêm vào một đoạn văn vào code source để giải thích là phải làm gì ở đó, dòng này có nhiệm vụ gì, kí hiệu này cho mục đích gì ..v.v..

Đó thật sự là một điều không thể thiếu vì kể cả những thiên tài về lập trình cũng cần phải thêm vào các chú thích ở đây hay ở kia. Những ghi chú này sẽ giúp bạn có thể:

- Dễ dàng đi vào trọng tâm của những gì bạn đã viết. Vì ta có thể dễ dàng quên mất nguyên tắc hoạt động chương trình mà bạn đã viết. 🤔 Bạn có thể mất nhiều ngày để suy nghĩ lại điều đó, bạn sẽ cần những chú thích của bản thân bạn để có thể tự hiểu lại ý nghĩa của việc mình làm.
- Nếu bạn đưa mã nguồn của bạn cho một ai khác và nếu người đó không hiểu nhiều lắm về nguyên tắc hoạt động chương trình của bạn, thì những ghi chú đó sẽ giúp họ làm quen nhanh hơn.
- Cuối cùng, cái đó cho phép tôi có thể thêm những chú thích vào những đoạn mã trong bài học khi hướng dẫn cho bạn. Điều đó giúp tôi giải thích cho bạn tốt hơn về tác dụng của những dòng code.

Có nhiều cách để thêm vào một lời chú thích. Tất cả phụ thuộc vào chiều dài của lời chú thích mà bạn muốn viết:

- Nếu **ngắn**: chỉ gồm 1 dòng, hoặc vài từ. Trong trường hợp đó bạn đánh vào double slash (//) sau đó là chú thích của bạn.

Ví dụ:

C code:

```
// Đây là một chú thích ngắn.
```

hoặc

```
printf("Xin chào"); // instruction này hiện thị lên màn hình "Xin chào"
```

- Nếu lời chú thích của bạn **dài**: bạn có nhiều cái để thuật lại, bạn cần viết rất nhiều câu và trên rất nhiều dòng. Trong trường hợp này :
 - i. Để mở đầu lời chú thích: hãy đánh một slash sau đó đánh dấu sao (/*)
 - ii. Để kết thúc: Đánh dấu sao rồi sau đó là slash (*/)

Ví dụ:

C code:

```
/* Đây là một chú thích gồm nhiều dòng */
```

Trở lại với chương trình hiển thị “Xin chào”, và thêm vào những lời chú thích để luyện tập:

C code:

```
/*  
Sau đây là những preprocessor directives.  
Nhưng dòng này cho phép thêm một số file vào project của bạn, nhưng file này thường được chúng  
ta gọi tên là “thu vien”  
Nho vào các file thu vien, chúng ta luôn có những hàm sẵn sàng làm việc.  
ví dụ như hàm printf: hiển thị một đoạn văn lên màn hình  
*/  
#include <stdio.h>  
#include <stdlib.h>  
/*  
Sau đây là function chính của chương trình tên là “main”. Nho function này mà chương trình của  
bạn có thể bắt đầu  
Chương trình này sẽ hiển thị “Xin chào” lên màn hình, đưa chương trình vào trạng thái pause,  
kết thúc  
*/  
int main(int argc, char *argv[])  
{  
    printf("Xin chào"); // instruction này hiển thị “Xin chào” lên màn hình  
    return 0; // Chương trình trả về giá trị 0 và kết thúc  
}
```

Trên đây là một chương trình với những dòng chú thích 😊

Khi ta biên dịch chương trình, tất cả những chú thích sẽ được bỏ qua, máy tính sẽ không đọc các dòng này. Những chú thích sẽ không xuất hiện khi ta chạy chương trình, chúng chỉ dành cho những người lập trình.

Bình thường thì ta không ghi chú ở mỗi dòng code của chương trình. Tôi đã nói rằng viết chú thích trong code source là một điều quan trọng nhưng chúng ta cần biết khi nào cần dùng đến, vì chú thích từng dòng như vậy sẽ tốn thời gian vô ích.

VD như khi mọi người đã biết rằng printf là hàm hiển thị một tin nhắn lên màn hình, bạn không cần phải chú thích thêm nữa về tác dụng của nó mỗi lần lập trình. 😊

Tốt hơn là bạn hãy chú thích nhiều cái trong một lần, chẳng hạn như để giải thích ý nghĩa của một dãy instruction nào đó, nó sẽ được sử dụng vào việc gì.

Và người lập trình chỉ cần ngó qua những lời chú thích, họ sẽ tự hiểu lấy toàn bộ.

Nắm vững: Những lời chú thích hướng dẫn người lập trình trong code source, nó cho phép chúng ta nhận ra nó, vì vậy hãy tập chú thích từng nhóm cùng lúc hơn là bạn chú thích cho từng dòng.

Và để kết thúc bài học này, tôi xin trích dẫn **một luật của IBM**:

Nếu đọc những chú thích mà bạn không hiểu chương trình hoạt động thế nào, hãy xóa bỏ tất cả.

Như bạn nhận thấy, chúng ta vẫn chưa hoàn toàn kết thúc hết toàn bộ bài học.

Và đây cũng là lần đầu tiên bạn thấy thế nào là mã lập trình thật sự, các từ ngữ, các kí hiệu, có thể khiến đầu óc hơi choáng váng một tí.

Thật ra điều đó cũng bình thường thôi, tất cả ai cũng đều như vậy trong lần đầu tiên. 😊

Trước khi bạn bước sang một giai đoạn mới, bạn hãy test lại những gì bạn đã biết.

Tôi cố tránh việc dạy bạn nhiều thứ trong một lúc, đơn giản là bạn sẽ không lãnh ngộ được gì cả nếu bạn học một cách quá nhanh và nhồi nhét.

Và tôi xin báo trước cho bạn biết, trong các phần tiếp theo sẽ có rất nhiều điều mới lạ mà bạn chưa biết.

TRẮC NGHIỆM KIẾN THỨC.

| |
|--|
| <p>❓ Một dòng preprocessor directives được bắt đầu bởi</p> <p>A. #</p> <p>B. {</p> <p>C. //</p> |
| <p>❓ Tên của function chính trong chương trình là ?</p> <p>A. printf</p> <p>B. master</p> <p>C. main</p> |
| <p>❓ Thư viện là gì?</p> <p>A. Những file source đã được viết trước đó gồm các function luôn sẵn sàng chờ bạn gọi ra.</p> <p>B. Một file cho phép bạn viết một đoạn văn lên màn hình</p> <p>C. Một nơi để ta có thể mượn những quyển sách về khoa học viễn tưởng</p> |
| <p>❓ Một instruction luôn được kết thúc bởi kí tự nào ?</p> <p>A. /*</p> <p>B. ;</p> <p>C. }</p> |
| <p>❓ Tên của hàm cho phép hiển thị một đoạn văn lên màn hình ?</p> <p>A. printf</p> <p>B. print</p> <p>C. afficher</p> |
| <p>❓ Kí tự nào cho phép ta xuống dòng khi hiển thị tin nhắn lên màn hình console?</p> <p>A. \t</p> <p>B. \n</p> <p>C. Chỉ đơn giản là nhấn phím enter để xuống dòng.</p> |
| <p>❓ Chú thích chỉ dành cho một dòng bắt đầu bởi :</p> <p>A. /*</p> <p>B. */</p> <p>C. //</p> |

Đáp án:

- 1 – A
- 2 – C
- 3 – A
- 4 – B
- 5 – A
- 6 – B
- 7 – C