

Bài 4: Thế giới của các biến số (variable)

Đây là một chương quan trọng, và bạn cần phải tập trung nhiều (Nói cách khác, đây không phải thời điểm để bạn phân tích đường bay của một con ruồi đang quanh quẩn bên cạnh). 😊

Tóm tắt lại những gì đã học:

Ở những bài trước trước, bạn đã được học cách để tạo một project mới trên IDE Code::Blocks. Tôi đã đặc biệt giải thích với bạn rằng việc tạo ra một chương trình trên cửa sổ khá phức tạp (và tôi cũng không nói với bạn về việc tạo ra một game 3D chơi trên mạng 😊).

Chúng ta bắt đầu học lập trình với việc học cách làm việc trên console. Và chúng ta đã học những điều khá hay ho như việc hiển thị một tin nhắn lên màn hình.

Tôi biết rằng bạn sắp sửa hứa với tôi rằng cái đó chưa giúp bạn điều gì cả. 😊

Và tại thời điểm này, bạn vẫn chưa biết cách làm thế nào gọi ra một **biến số**, thứ mà tất cả những ngôn ngữ lập trình như C đều bắt buộc phải sử dụng.

Nào chúng ta hãy nói về nó !

❓ Vậy thì chính xác biến số là gì ?

Tôi sẽ giải thích tất cả về nó trong phần này, bạn sẽ không phải chờ đợi lâu đâu, nhìn một cách tổng quát chúng ta sẽ học cách đưa những con số vào trong bộ nhớ của máy tính.

Tôi sẽ bắt đầu với những lời giải thích về bộ nhớ của máy tính, nguyên tắc hoạt động, máy tính có thể nhớ bao nhiêu thứ khác nhau?

Vấn đề này có thể đơn giản đối với một số người, nhưng bài giảng của tôi chỉ dành cho những người vẫn chưa biết bộ nhớ máy tính là gì.

Phần này ta sẽ học :

Công việc của bộ nhớ:

- Khai báo một biến số
- Hiển thị giá trị một biến số
- Lưu lại giá trị được chọn
- TRẮC NGHIỆM KIẾN THỨC.

Công việc của bộ nhớ

Bài giảng này có mối liên hệ trực tiếp với bộ nhớ của máy tính.

Con người cũng như máy tính đều cần lưu giữ lại một số cái gì đó, con người chỉ có duy nhất bộ não nhưng trên máy tính thì có nhiều dạng bộ nhớ khác nhau.

❓ Tại sao máy tính cần nhiều loại bộ nhớ khác nhau? Một bộ nhớ duy nhất không đủ cho máy tính, có phải vậy không?

Không, thực tế người ta chỉ cần một bộ nhớ có **tốc độ lưu nhanh** và **khả năng chứa lớn** (để có thể lưu lại nhiều thứ quan trọng).

Nhưng cho đến thời điểm hiện tại, chúng ta vẫn chưa tạo được những bộ nhớ giống như vậy. Vì các bộ nhớ nhanh thì đắt tiền nên các bộ nhớ được tổ chức thành nhiều cấp, cấp có dung lượng ít thì nhanh nhưng đắt tiền hơn cấp có dung lượng cao hơn. Những bộ nhớ có tốc độ lưu càng nhanh sẽ có dung lượng càng nhỏ.

Vậy máy tính của chúng ta được lắp đặt gồm:

- Những bộ nhớ có tốc độ lưu nhanh nhưng khả năng chứa nhỏ.
- Những bộ nhớ có tốc độ lưu chậm nhưng khả năng chứa lớn hơn rất nhiều.

Bạn vẫn theo kịp tôi chứ 😊

Những khác biệt về bộ nhớ:

Để cho bạn dễ hiểu, đây là những loại bộ nhớ khác nhau có trong một máy tính được sắp xếp từ nhanh đến chậm:

1. *Registers*: Bộ nhớ cực nhanh được đặt trực tiếp trong bộ xử lý của máy tính (processor).
2. *Memory cache*: Làm cầu nối giữa registers và RAM.
3. *Main memory (RAM)*: Là một bộ nhớ mà chúng ta sử dụng thường xuyên nhất.
4. *Ổ cứng (Hard Disk Drive)*: Cái này các bạn biết đến nhiều nhất, người ta thường lưu trữ dữ liệu ở đây.

Những registers chỉ có thể chứa được một vài số, trái ngược hẳn với ổ cứng có thể chứa một số lượng lớn các tập tin.

⚠ Khi tôi nói một bộ nhớ “chậm” là đang dựa theo thang đo máy tính, 8 phần nghìn giây để vào đến ổ cứng thật sự là quá lâu!

❓ Có cần phải nắm tất cả những điều này?

Từ bây giờ, các bạn sẽ học về lập trình, và các bạn thường chỉ làm việc trên RAM nên các bạn cần biết đôi chút về nó. Chúng ta sẽ tìm hiểu cách đọc và lưu các tập tin lên ổ cứng (nhưng có lẽ là trong các bài học sau). Còn về Memory cache và registers thì không cần phải chạm đến vì máy tính của bạn sẽ tự làm việc đó.

⚠ Trong ngôn ngữ lập trình bậc thấp, như assembler (viết tắt của "ASM"), một ngôn ngữ tôi đã từng sử dụng, chúng ta phải làm việc trực tiếp với registers, việc làm một phép toán nhân đơn giản thật sự là cả một quá trình chiến đấu gian nan! May mắn là việc đó trên C (và trên nhiều ngôn ngữ lập trình khác) thực hiện đơn giản hơn rất nhiều.

Cần phải nói thêm một điều quan trọng cuối cùng: chỉ có ổ cứng giữ lại tất cả những gì mà nó chứa. Tất cả các bộ nhớ khác (registers, Memory cache, RAM) đều là những bộ nhớ nhất thời: khi mà bạn tắt máy tính đi thì tất cả dữ liệu trong đó sẽ mất đi.

May mắn là dữ liệu trong ổ cứng của bạn vẫn không đổi để nhắc nhở máy tính của bạn ở tình trạng nào khi bật lên. 😊

Hình ảnh của RAM:

Chúng ta sắp sửa làm việc với RAM, tôi nghĩ rằng tôi nên giới thiệu nó với bạn 😊

Đây là máy tính của bạn:

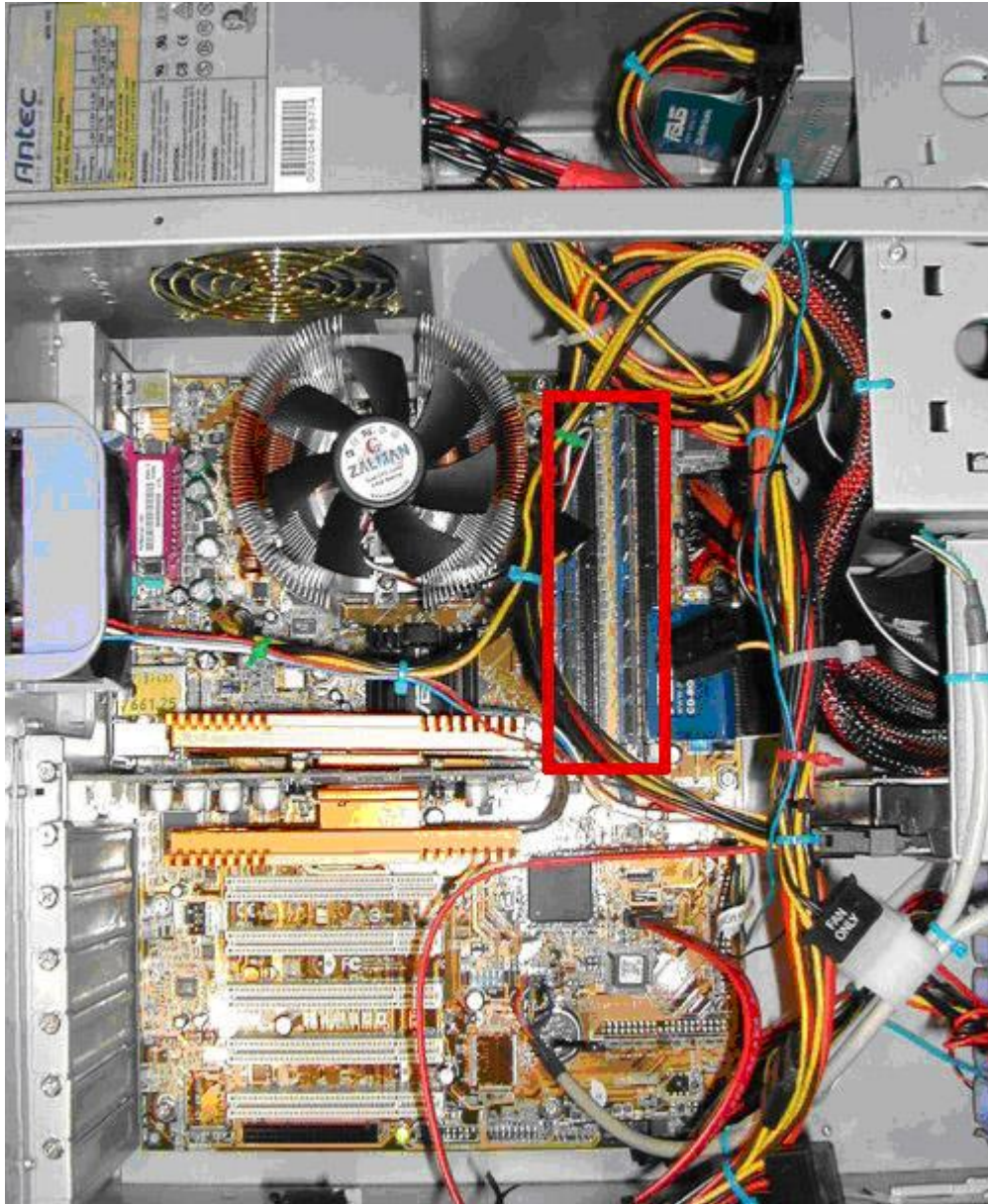


Các bạn đã biết thế nào là bàn phím, chuột, màn hình và thùng máy.

Bây giờ chúng ta chỉ quan tâm đến thùng máy của bạn, trung tâm của máy tính, nó chứa tất cả các loại bộ nhớ:



Cái mà ta đang hứng thú tìm hiểu nằm bên trong thùng máy, khi mở ra:



Bạn có cảm thấy thích nó không? 🤔

Các bạn hãy yên tâm, tôi sẽ không yêu cầu các bạn phải biết chúng hoạt động như thế nào, tôi chỉ muốn bạn biết chỗ để tìm thấy RAM trong thùng máy, nó nằm trong ô chữ nhật màu đỏ.

Tôi sẽ không chỉ ra những bộ nhớ khác (registers và cache) nằm ở đâu vì nó khá nhỏ để có thể thấy được bằng mắt của chúng ta. 😊

Và đây là hình dáng thật sự của RAM:



Biểu đồ của RAM:

Nếu ta nhìn một cách bình thường trên RAM thì chẳng thấy được gì cả. Nhưng, điều quan trọng là cần biết bên trong nó hoạt động như thế nào. Đây chính là điều tôi muốn hướng dẫn các bạn.

Tôi sẽ vẽ cho các bạn một biểu đồ về cách hoạt động của RAM, nó cực kì đơn giản. 😊 Nếu bạn nắm được biểu đồ này thì điều đó vô cùng tốt đối với bạn. 😊

Địa chỉ	Giá trị
0	145
1	3.8028322
2	0.827551
3	3901930
...	...
3 448 765 900 126	940.5118

Biểu đồ hoạt động của RAM

Như bạn thấy, nó được chia làm 2 cột:

- Một cột **địa chỉ (address)**: địa chỉ là một số cho phép máy tính có thể xác định vị trí trong RAM. Nó bắt đầu từ địa chỉ 0 và kết thúc ở địa chỉ 3 448 765 900 126... Hic, tôi không hề biết rõ số lượng địa chỉ chứa trong RAM, tôi chỉ biết rằng nó có rất nhiều. Bởi vì nó phụ thuộc vào dung lượng bộ nhớ mà bạn có. Chỉ có thể nói là, bạn có RAM, bạn có thể để vào đó nhiều thứ. 😊
- Và mỗi địa chỉ chứa một **giá trị** (một số, value): Máy tính của bạn đưa vào RAM những số này để có thể nhớ ngay lập tức. Và người ta chỉ có thể đưa vào một số cho một địa chỉ trong RAM!

Và RAM không thể chứa gì khác ngoài những con số.

❓ **Vậy làm cách nào để chúng ta có thể lưu lại những chữ cái ?**

Đó là một câu hỏi thú vị, trên thực tế, đối với máy tính thì những chữ cái cũng là những con số!

Một câu văn chính là một dãy những con số !

Có một bảng viết về sự tương ứng giữa chữ cái và số (bảng mã ASCII), ví dụ số 67 tương ứng với chữ Y, tôi không nói nhiều về vấn đề này, nếu có cơ hội chúng ta sẽ tìm hiểu về nó sau.

Trở lại với biểu đồ của chúng ta. Hãy xem xét một vấn đề đơn giản: nếu máy tính muốn lưu lại giá trị 5 (có thể là số mạng sống của nhân vật mà bạn chơi trong game nào đó), nó sẽ đặt số 5 vào một vị trí nào đó trong bộ nhớ. (Ví dụ tại địa chỉ 3 062 199 902).

Sau đó, khi muốn tìm lại giá trị này, máy tính sẽ đến “ô” bộ nhớ n° 3 062 199 902, tại đó nó tìm thấy 5 !

Và đó là nguyên tắc hoạt động của bộ nhớ, có thể bạn vẫn còn một chút mập mờ (Đâu là lợi ích của việc đặt một số vào một địa chỉ của bộ nhớ?), bạn sẽ hiểu rõ hơn vấn đề này ở những phần sau của bài hướng dẫn. 😊

Cách khai báo một biến số

Bạn hãy tin tôi rằng một ít giới thiệu về bộ nhớ sẽ rất tiện lợi và tốt hơn cho bạn, giúp bạn có thể tưởng tượng dễ dàng hơn.

Nhưng bây giờ chúng ta phải biết làm cách nào để sử dụng nó. 😊

Vậy thế nào là một biến số ?

Chỉ đơn giản là một thông tin nhỏ được lưu trữ trong RAM.

Chúng ta gọi nó là « biến số » vì nó có thể thay đổi trong quá trình thực hiện chương trình. Ví dụ, số 5 vừa rồi của chúng ta có khả năng bị giảm đi (khi mà nhân vật bạn chết thì mạng sống sẽ giảm xuống). Khi mà giá trị này tiến đến 0 thì trò chơi sẽ kết thúc, game over.

Các bạn sẽ thấy chương trình của chúng ta sẽ chứa đầy những biến số.

Trên ngôn ngữ C, một biến số có 2 thành phần:

- Một **giá trị**: đó là số mà nó chứa, ví dụ như 5.
- Một **tên gọi**: tên gọi này sẽ giúp ta nhận ra nó. Trên ngôn ngữ C, chúng ta không cần phải nhớ địa chỉ của biến số, chúng ta chỉ cần chỉ ra tên của biến số. Và bộ dịch (Compiler) sẽ thực hiện việc chuyển đổi giữa chữ và số.

Gọi tên một biến số:

Trong ngôn ngữ của chúng ta, biến số chỉ số mạng sống của nhân vật trong một trò chơi điện tử nào đó thường được gọi là “mạng sống nhân vật”, hoặc một tên nào khác cùng loại.

Trong ngôn ngữ C, mỗi biến số có một tên gọi, nhưng không phải muốn đặt tên thế nào tùy theo ý thích của bạn cũng được đâu. Dưới đây là một số nguyên tắc khi đặt tên cho biến số:

- Chúng ta chỉ có thể đặt tên nó bằng những chữ cái viết thường hay viết hoa và những con số (abcABC012...).
- Tên của biến số phải **bắt đầu bằng một chữ cái**. Chúng ta **không được sử dụng khoảng trống** « » , thay vào đó chúng ta có thể sử dụng **kí tự** « _ » (underscore). Đó là kí tự duy nhất không thuộc dạng chữ cái hay số được phép sử dụng.
- Bạn cũng không được phép sử dụng chữ cái mang dấu trọng âm. (ví dụ eèèèa...).

Và một điều hết sức quan trọng mà bạn cần phải nắm đó là trong ngôn ngữ C (C++ cũng như thế) **có sự khác nhau giữa chữ thường và chữ hoa**: chieu_rong, CHIEU_RONG và CHieu_RoNg là tên của 3 biến số khác nhau trong ngôn ngữ C. Đối với chúng ta thì chúng có vẻ hoàn toàn giống nhau! Và đây là các biến số được đặt tên chính xác: mangsongNhanvat, mangsong_nhanvat, ho, ten, so_dien_thoai, sodidong.

Mỗi người có cách thức gọi tên biến số khác nhau. Trong phần này, tôi giới thiệu cho bạn cách thức gọi tên biến số của riêng tôi:

- Tên của biến số, tôi luôn bắt đầu bằng chữ cái thường.
- Nếu tên của biến số gồm nhiều chữ, thì mỗi chữ tôi sẽ viết hoa ở kí tự đầu tiên

Tôi thích bạn thực hiện giống như tôi, vì điều đó giúp chúng ta có thể làm việc dễ dàng với nhau.



Bạn hãy đặt cho biến số những tên gọi rõ ràng. Chúng ta có thể rút ngắn tên của mangsong_NhanVat bằng ms_NV. Điều đó có thể giúp tên gọi ngắn hơn, nhưng không hề rõ ràng khi bạn viết chương trình. Bạn đừng ngại việc đặt tên dài cho biến số vì điều đó sẽ giúp chương trình của bạn dễ đọc, dễ hiểu hơn.

Những dạng của biến số:

Các bạn có thể xem máy tính không khác gì một cỗ máy lớn dành cho công việc tính toán, nó không biết gì khác hơn ngoài những con số.

Và tôi có một tin đặc biệt là **có nhiều dạng biến số !**

Ví dụ, có những số tự nhiên dương:

- 45
- 398
- 7650

Cũng có những số thực:

- 75,909
- 1,7741
- 9810,7

Hơn nữa cũng có những số nguyên âm:

- -87
- -916

Và những số thực âm:

- -76,9
- -100,11

Và bạn sẽ thấy phần lớn chúng ta chỉ sử dụng những số tự nhiên vì nó dễ dàng sử dụng. 😊



Hãy chú ý với những số thực! Máy tính của bạn không hiểu dấu phẩy là gì đâu, chúng ta chỉ sử dụng dấu chấm. Bạn không thể viết 54,9, thay vào đó là 54.9!

Và không chỉ như vậy! Đối với những biến số dạng số tự nhiên (char, int, long), còn có thêm các loại đặc biệt khác mang tên « unsigned » (không có dấu), tại đó chúng ta chỉ có thể đưa vào những số tự nhiên. Để sử dụng, chỉ cần đặt « unsigned » ở phía trước :

unsigned char	0 đến 255
unsigned int	0 đến 4 294 967 295
unsigned long	0 đến 4 294 967 295

Như bạn đã thấy, những biến dạng unsigned không thể chứa những số âm, nhưng nó có lợi thế là mở rộng giới hạn chứa những số dương lên gấp đôi (ví dụ: signed char có giới hạn 128, trong khi đó unsigned char có giới hạn 255).



Bạn cần lưu ý rằng dạng biến số char nên được khai báo hoặc có signed, hoặc unsigned, không nên đứng một mình. Lý do đơn giản là dạng biến số này sẽ có dấu hay không dấu tùy vào các loại máy tính khác nhau. Trước khi khai báo một biến số, hãy suy nghĩ dạng biến số nào bạn sẽ cần dùng đến.

? Tại sao phải tạo ra 3 dạng biến số cho những số tự nhiên như vậy? Chúng ta chỉ cần 1 dạng là đủ rồi mà, không phải vậy sao?

Người ta tạo nhiều dạng biến số khác nhau như thế để tiết kiệm bộ nhớ. Khi mà chúng ta bảo máy tính rằng chúng ta cần một biến số dạng “char”, thì máy tính sẽ sử dụng bộ nhớ ít hơn khi chúng ta bảo rằng cần bộ nhớ dạng “long”.

Việc này sẽ có ý nghĩa trong giai đoạn bộ nhớ máy tính còn nhiều giới hạn. Ngày nay, RAM máy tính đã tiên tiến hơn rất nhiều nên việc này không còn là vấn đề thật sự nữa. Chúng ta không cần nghĩ nhiều đến việc chọn dạng biến số nào để sử dụng. Nếu biến số của bạn có nhu cầu nhận một giá trị tương đối lớn thì hãy nghĩ đến việc sử dụng long.

Tôi nói nghiêm túc rằng bạn không cần phải suy nghĩ nhiều lắm về cách chọn dạng biến số trong thời điểm hiện tại. 😊

Chúng ta chỉ cần phân biệt sự khác biệt giữa dạng số nguyên và số thực:

- Đối với số tự nhiên, người ta thường dùng int.
- Đối với số thực, người ta thường dùng double.

Khai báo một biến số

Cuối cùng chúng ta cũng đến được đây, và bây giờ bạn hãy tạo một project mới lấy tên là “variables”.(biến số)

Bạn sẽ thấy làm cách nào để chúng ta khai báo một biến số, hay nói cách khác là bạn sẽ **yêu cầu quyền sử dụng một ít bộ nhớ của máy tính**.

Bạn chỉ cần làm theo trình tự sau:

1. Chỉ ra dạng của biến số cần tạo.
2. Nhấn phím spacebar để cách khoảng.
3. Chỉ ra tên của biến số cần tạo.
4. Cuối cùng là chấm phẩy ; đừng quên điều đó.

Ví dụ nếu như tôi muốn khai một biến số mangsongNhanVat, tôi sẽ làm như sau:

C code:

```
int mangsongNhanVat;
```

Chỉ đơn giản vậy thôi! 😊

Và một vài ví dụ khá ngu khác :

C code:

```
int diemToan;  
double tongChiPhiNhanDuoc;  
unsigned soluongNguoiChuanBiXemTenCuaMotBienSoKhaLaDai;
```

Tôi nghĩ bạn cũng đã hiểu được nguyên tắc của nó rồi. 😊

Việc chúng ta vừa làm gọi là **variable declaration (khai báo biến số)**, hãy nắm vững thuật ngữ này 😊

Bạn phải thực hiện việc khai báo biến số ở vị trí bắt đầu của các function. Và trong thời điểm này chúng ta chỉ có được duy nhất một function (function *main*), bạn hãy khai báo biến số như sau:

C code:

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

int main(int argc, char *argv[]) // Tương tự với int main( )
{
    int mangsongNhanVat;
    return 0;
}
```

Nếu bạn thực hiện việc dịch và chạy chương trình vào lúc này thì chắc hẳn bạn sẽ ngạc nhiên rằng chẳng có gì xảy ra cả 😊

Giải thích

Trước khi bạn cho rằng tôi đùa với bạn thì hãy nghe tôi giải thích vài lời. 😊

Thực tế đã nó một vài thứ diễn ra nhưng bạn không thể nào thấy được. Khi mà chương trình chạy đến đoạn mã khai báo biến số của bạn, nó chỉ yêu cầu máy tính một cách lịch sự rằng nó sẽ sử dụng một ít khoảng trống trong RAM của máy tính.

Nếu không có vấn đề gì, máy tính sẽ trả lời « dùng đi, tự nhiên như ở nhà mày vậy ».

⚠️ Vấn đề chỉ xảy ra khi bộ nhớ của bạn không còn khoảng trống nữa. May mắn là điều này vô cùng khó xảy ra nếu sử dụng những biến số dạng `int` để làm đầy bộ nhớ của máy tính. 😊

Và biến số của bạn đã được tạo ra một cách hoàn hảo.

⚠️ Có một điều bạn cần biết: nếu bạn có nhiều biến số cần khai báo và các biến số này cùng một dạng, bạn không cần thiết phải khai báo mỗi biến số cho mỗi dòng. Bạn chỉ cần phân biệt các biến số bởi những dấu phẩy trên cùng một dòng :

C code:

```
int mangsongNhanVat, capdoTroChoi, capdoNhanVat;
```

Đoạn code này đã khai báo 3 biến số dạng `int` cho các biến số `mangsongNhanVat`, `capdoTroChoi`, `capdoNhanVat`.

Và bây giờ ?

Sau khi đã khai báo xong biến số, chúng ta có thể đưa cho chúng những giá trị. 😊

Đưa giá trị vào biến số

Không có gì là khó khăn, nếu bạn muốn cho biến số mangsongNhanVat một giá trị, bạn chỉ cần làm như sau :

C code:

```
mangsongNhanVat = 5;
```

Vậy là xong rồi, bạn không cần làm thêm điều gì khác. Bạn chỉ cần đặt tên của biến số, cho một dấu bằng, kế tiếp là giá trị bạn muốn đặt vào nó. Ở đây chúng ta cho mangsongNhanVat giá trị 5.

Dưới đây là chương trình hoàn thiện:

C code:

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

int main(int argc, char *argv[])
{
    int mangsongNhanVat;
    mangsongNhanVat = 5;

    return 0;
}
```

Và khi chạy chương trình thì màn hình vẫn chưa có gì thay đổi, nó chỉ diễn ra trong bộ nhớ.

Tại một ô bộ nhớ nào đó trong máy tính, giá trị 5 đã được đặt vào. Tuyệt vời đúng không? 😄

Và hay hơn nữa là:

C code:

```
int mangsongNhanVat;
mangsongNhanVat = 5;
mangsongNhanVat = 4;
mangsongNhanVat = 3;
```

Trong ví dụ này, khi chạy chương trình, biến nhận giá trị 5 đầu tiên, sau đó là 4 và cuối cùng là 3. Việc này diễn ra vô cùng nhanh trên máy tính, chương trình kết thúc khi bạn chưa kịp chớp mắt xong 😊

Giá trị của một biến số mới

Đây là một câu hỏi khá quan trọng mà tôi muốn nhấn mạnh :

❓ Khi mà ta khai báo một biến, thì nó sẽ nhận giá trị nào đầu tiên ?

Thực tế, khi mà máy tính bạn đọc dòng này :

C code:

```
int mangsongNhanVat;
```

Đồng ý là biến sẽ chiếm 1 vị trí trong bộ nhớ của RAM. Nhưng giá trị của biến số lúc này là bao nhiêu ? Là 0 lúc khởi đầu đúng không ?

Câu trả lời là không. Không, không và không. Không có giá trị nào lúc khởi đầu cả. Bộ nhớ sẽ giành chỗ cho biến số nhưng tại vị trí đó, giá trị sẽ không đổi. Máy tính sẽ không hề xóa những gì đã được đặt vào trước đó (có thể vị trí đó đã được dùng cho một chương trình cũ từng chạy trên máy tính trước đây)

Nếu vị trí này vẫn chưa sử dụng qua lần nào thì có thể nó sẽ mang giá trị là 0. Nhưng nếu một chương trình nào khác đã sử dụng qua rồi thì nó có thể mang giá trị là 368, 18 hay một số nào khác bất kỳ.

Chúng ta cần phải chú ý kỹ vấn đề này để tránh các sai sót về sau. Tốt nhất bạn hãy gán cho nó giá trị ngay sau khi vừa khai báo xong. Trình biên dịch có thể hiểu được nếu như ta khai báo và gán giá trị một biến số trong cùng một lúc:

C code:

```
int mangsongNhanVat = 5;
```

Như trên, biến số mangsongNhanVat đã được khai báo và nhận tức khắc giá trị 5.

Lợi ích của việc này là bạn luôn chắc rằng biến số đó luôn nhận giá trị chính xác như bạn muốn.

Những constants (hằng số)

Đôi khi chúng ta cần sử dụng những giá trị không đổi trong suốt quá trình sử dụng chương trình. Có nghĩa là sau khi khai báo, biến số sẽ nhận một giá trị mà không cách nào có thể thay đổi được.

Những biến số này chúng ta gọi là các **constants** (hằng số), lý do là giá trị mà nó mang sẽ luôn được giữ nguyên như thế.


Để khai báo một constant, ta làm như sau: chúng ta thêm từ « const » trước dạng biến số mà bạn khai báo.

Mặt khác, chúng ta bắt buộc phải gán cho nó một giá trị ngay trong thời điểm bạn khai báo nó. Giống như cách mà ta đã thấy vừa rồi. Sau đó, bạn không thể nào thay đổi giá trị đó nữa, vì mọi thứ đã được qui định xong hết rồi.

Ví dụ về cách khai báo một constants:

C code:

```
const MANGSONG_NHANVAT KHOIDAU = 5;
```

 Việc tôi chỉ sử dụng những chữ cái in hoa để đặt tên cho constants là không bắt buộc. Làm như thế để giúp tôi có thể dễ dàng phân biệt những biến số với những constants. Ghi thêm rằng tôi vẫn sử dụng dấu underscore _ vào vị trí của khoảng trống « ».

Sau đó, bạn có thể sử dụng constants như một biến số bình thường. Khác biệt duy nhất là nếu bạn thử thay đổi giá trị của nó sau đó và thực hiện dịch chương trình thì compiler sẽ báo lỗi. 😊

Tôi gọi nó là « death zone » (hay là vùng chết). Trong trường hợp đó, compiler sẽ hiển thị lên màn hình: [Warning] assignment of read-only variable 'MANGSONG_NHANVAT_KHOIDAU' (Dịch ra: “bạn thật là ngu ngốc, tại sao bạn lại cố gắng thay đổi giá trị của một constant chứ?”)

Hiển thị giá trị của biến số

Chúng ta đã biết cách hiển thị một đoạn văn với function *printf*.

Bây giờ, chúng ta sẽ xem làm sao để hiển thị một giá trị của biến số cũng với function này.

Chúng ta cũng sẽ sử dụng *printf* với phương pháp cũ, nhưng thêm vào một kí tự đặc biệt tại vị trí mà chúng ta muốn giá trị của biến số đó.

Ví dụ :

C code:


```
printf ("Ban con %d hoisinh");
```

Kí tự đặc biệt mà tôi đã nói với bạn đó là một « % » sau đó là những chữ cái « d ». Những kí tự này cho phép chúng ta hiển thị dạng của biến số.

« d » có nghĩa là tôi muốn hiển thị một số dạng int.

Còn rất nhiều kí tự đặc biệt khác có thể sử dụng. Nhưng để dễ dàng, lúc này bạn chỉ cần nắm những loại sau:

Format	Type
"%d"	int
"%ld"	long
"%f"	float
"%lf"	double

 Cần lưu ý rằng format dùng để hiển thị một float và một double là giống nhau.

Tôi sẽ nói cho bạn biết những kí tự đặc biệt khác về sau.

Chúng ta sắp xong rồi, chúng ta đã chỉ ra vị trí cần hiển thị một số, nhưng chúng ta vẫn chưa nói là hiển thị số nào. Vì thế chúng ta cần chỉ cho function *printf* biết phải hiển thị biến số nào.

Bằng cách đánh tên của biến số đó sau khi thêm vào một dấu phẩy sau khi kết thúc dấu '' giống như sau:

C code:

```
printf ("Ban con %d hoisinh", mangsongNhanVat);
```

%d sẽ thay thế bởi biến số mà ta đã chỉ ra sau dấu phẩy, trường hợp này là mangsongNhanVat.

Chúng ta thử chạy chương trình nhé ?

<-----

C code:

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

int main(int argc, char *argv[])
{
    int mangsongNhanVat = 5; // Khoi dau, ban co 5 lan hoi sinh
    printf ("Ban co %d lan hoi sinh\n", mangsongNhanVat);
    printf ("***** B U M *****\n"); // Ban bi trung mot phat sung vao dau
    mangsongNhanVat = 4; // Ban bi mat mot lan hoi sinh !
    printf ("Xin chia buon, ban chi con %d lan hoi sinh !\n\n", mangsongNhanVat);

    return 0;
}
```

Cái này gần giống như một game điện tử rồi (bạn hãy tưởng tượng nhiều một tí)
Chương trình hiển thị cái này ra màn hình:

Console:

```
Ban co 5 lan hoi sinh
***** B U M *****
Xin chia buon, ban chi con 4 lan hoi sinh !

Press any key to continue.
```

Bạn phải hiểu được điều gì diễn ra trong chương trình của bạn :

1. Đầu tiên, nhân vật có 5 mạng sống, chúng ta hiển thị bằng printf
2. Sau đó, « bùm » nhân vật trúng phải một phát súng vào đầu.
3. Cuối cùng, nhân vật còn 4 mạng sống, chúng ta hiển thị bằng printf.


Đơn giản là như vậy.

Hiển thị nhiều biến số trong một function printf

Chúng ta luôn có thể hiển thị giá trị của nhiều biến số chỉ trong một function *printf* duy nhất. Chỗ này sẽ hiển thị %ld và chỗ kia hiển thị %lf, tùy theo bạn muốn, sau đó chỉ ra theo thứ tự lần lượt những biến số tương ứng, cách nhau bởi những dấu phẩy.

Ví dụ :

```
printf ("Ban co %d lan hoi sinh va dang o man chơi thu %d", mangsongNhanVat, capdo);
```

 Hãy chỉ ra những giá trị của bạn theo đúng thứ tự. %d đầu tiên sẽ thay thế bằng biến số đầu tiên (mangsongNhanVat), và %d thứ hai sẽ thay thế bởi biến số thứ hai (capdo). Nếu như bạn nhầm lẫn vị trí, những gì bạn muốn hiển thị sẽ không còn đúng nữa.

Và chúng ta hãy thử test lại một tí, ghi thêm rằng trong đoạn code bên dưới tôi không ghi những dòng ở trên cùng (những preprocessor directives bắt đầu bởi những #), và tôi sẽ **giả định rằng bạn sẽ luôn thêm nó vào ở đầu chương trình**.

C code:

```
int main(int argc, char *argv[])
{
    int mangsongNhanVat = 5, capdo = 1;

    printf ("Ban co %d lan hoi sinh va ban dang o man chơi thu %d\n", mangsongNhanVat, capdo);

    return 0;
}
```

Và nó sẽ hiển thị:

Console:

```
Ban co 5 lan hoi sinh va ban dang o man chơi thu 1
```

Cách gán giá trị cho biến số

Những biến số bắt đầu làm bài học này thú vị hơn rồi nhỉ. Chúng ta sẽ học cách yêu cầu người dùng nhập một số vào console. Số này sẽ được một biến số lưu lại. Một khi bạn thực hiện được điều này, bạn có thể làm thêm rất nhiều việc sau đó. 😊

Để yêu cầu người dùng đưa vào một cái gì đó vào trong console, chúng ta sẽ sử dụng một function khác, và function này đã có sẵn trong thư viện của bạn: function đó là *scanf*

Cách sử dụng *scanf* khá giống với *printf*. Bạn phải đặt %d hay %lf trong cặp dấu "..." để giải thích với máy tính rằng bạn muốn người dùng đưa vào một số nguyên hay một số thực. Sau đó bạn phải chỉ ra tên của biến số sẽ nhận lấy giá trị đó.

Bạn sẽ thấy điều đó trong ví dụ này :

C code:

```
scanf ("%d", &tuoi);
```

Chúng ta phải đặt %d (hoặc %lf) trong cặp dấu "..."
Một khác chúng ta thêm vào & trước tên của biến số.

❓ Vậy tại sao phải thêm & trước tên của biến số ?

Tôi sẽ không giải thích cho bạn tất cả ở đây. Nhưng hãy tin tôi, tôi sẽ giải thích cho bạn vấn đề này trong một bài khác sau này, tôi hứa đấy ! 😊

Trở lại, khi mà chương trình của bạn chạy đến *scanf*, nó sẽ dừng lại và đợi người sử dụng đưa vào một số. Số này sẽ được đưa vào biến số « *tuoi* ».



Hãy chú ý, có một vài sự khác nhau giữa *printf* và *scanf* ! Để gán một giá trị dạng float, ta dùng format "%f", nhưng để gán một giá trị dạng double ta dùng format "%lf"

Đây là một chương trình nhỏ yêu cầu biết tuổi của người sử dụng và nó sẽ hiển thị ra sau đó :

Code C:

```
int main(int argc, char *argv[])
{
    int tuoi = 0; // Khoi tao bien so gia tri la 0

    printf ("Ban bao nhieu tuoi?\n");
    scanf ("%d", &tuoi); // May tinh yeu cau nhap tuoi voi scanf
    printf ("Oh! tuoi cua ban la %d !\n\n", tuoi);

    return 0;
}
```

Console:

```
Ban bao nhieu tuoi?
20
Oh! tuoi cua ban la 20 !
```

Chương trình sẽ dừng lại và hiển thị « Ban bao nhieu tuoi? ». Dấu nháy sẽ xuất hiện trên màn hình. Các bạn phải đánh vào một số tự nhiên (tuổi của bạn). Sau đó nhấn Enter để xác nhận, và chương trình sẽ tiếp tục hoạt động.

Sau đó, chương trình sẽ hiển thị giá trị của biến số « tuoi » lên màn hình (“Oh! tuoi cua ban la 20 !”).

Nguyên tắc hoạt động là như vậy. 😊

Nhờ vào function *scanf* chúng ta có thể yêu cầu người sử dụng đưa ra một số thông tin cá nhân.

Viết thêm rằng bạn chỉ có thể đưa vào đó một số tự nhiên :

- Nếu bạn nhập vào đó một số thực, ví dụ như 2.9, nó sẽ tự động làm tròn, nghĩa là nó chỉ giữ lại phần nguyên. Trong trường hợp này số 2 sẽ được biến số lưu lại.
- Nếu bạn đánh vào bất kì một chữ cái nào đó («éèydf »), biến số sẽ không thay đổi giá trị. Điều này cũng tốt vì trước chúng ta đã gán cho biến số giá trị 0. Sau khi nhập những chữ cái vào thì ngay lập tức, chương trình hiển thị « 0 tuoi », chứng tỏ scanf không được thực hiện. Nếu sau khi khai báo biến số chúng ta không gán cho nó giá trị nào, chương trình bạn có thể hiển thị bất cứ cái gì !

Chúng ta sắp kết thúc bài học về các biến số 😊

Tôi xin nhắc lại là biến số sẽ được sử dụng thường xuyên khi lập trình. Nếu như bạn hiểu rằng biến số là một thông tin được đưa vào bộ nhớ tạm thời thì bạn đã hiểu bài giảng này. Không có điều gì khác ngoài việc bạn cần biết những dạng biến số (char, int, long, double...).

Hãy tự luyện tập cách hiển thị những biến số lên màn hình và cách nhập vào giá trị một biến số bằng bàn phím với scanf.

Trong chương tiếp theo, chúng ta sẽ học cách làm sao thực hiện các tính toán trên ngôn ngữ C. Yêu cầu bạn phải sử dụng tốt *printf* và *scanf*. 😊

TRẮC NGHIỆM KIẾN THỨC.

❓ Khi ta khai báo một biến số, bộ nhớ nào sẽ được sử dụng ?

- A. Registers
- B. Memory cache
- C. RAM (main memory)
- D. Hard Disk Drive

❓ Khi tắt máy tính, bộ nhớ nào sẽ không bị mất dữ liệu ?

- A. Registers
- B. Memory cache
- C. RAM (main memory)
- D. Hard Disk Drive

❓ Biến số nào không được đặt tên chính xác ?

- A. vitriMenu
- B. chieurongCửaSổ
- C. tuoi_Capital

❓ Dạng biến số nào có thể lưu trữ số 76.8 ?

- A. **char**
- B. **long**
- C. **double**
- D. **int**

❓ Dạng biến số nào có thể lưu trữ số -1000 ?

- A. **int**
- B. **unsigned int**
- C. **unsigned double**

❓ Nếu như biến số "taikhoanNganHang" thuộc dạng int có giá trị là 6 500 000 , màn hình sẽ hiển thị đoạn mã này thế nào ?

Code:

```
printf("Ban co %d dong trong tai khoan", taikhoanNganHang);
```

- A. **Ban co %d dong trong tai khoan**
- B. **Ban co 6 500 000 dong trong tai khoan**
- C. **Ban co d dong trong tai khoan, taikhoanNganHang**

Đáp án:

- 1- C
 - 2- D
 - 3- B
 - 4- C
 - 5- A
 - 6- B
-