Bài 9: Test Program

Trò chơi người treo cổ

Chỉ nói ko thôi thì chưa đủ: Chắc hẳn là các bạn hiểu rõ tầm quan trọng của việc thực hành đúng không? Tôi chắc chắn việc thực hành thật sự cần thiết đối với bạn, chúng ta vừa được học rất nhiều những khái niệm, lý thuyết … và bất cứ điều gì bạn được đọc, những gì bạn nói ra, bạn sẽ không bao giờ thực sự hiểu sâu về nó cho đến khi bạn bắt tay vào thực hành.

Trong lần học này, tôi đề nghị chúng ta sẽ làm ra trò chơi "Người treo cổ". Đây là một trò chơi cổ điển quen thuộc về các từ (bạn nào xài kim từ điển thì biết ngay trò này), bạn sẽ phải đoán những chữ cái bị ẩn trong các từ. Trò người treo cổ lần này sẽ được chơi theo kiểu của ngôn ngữ C trên màn hình console.

Mục đích chính của chúng ta là giúp bạn có thể nắm vững tất cả những kiến thức đã được học trước giờ. Những con trỏ (pointer), chuỗi ký tự (string), tập tin (file), mảng (array), cấu trúc (structure)… ok, tất cả sẽ ổn thôi.

Một số chỉ dẫn.

Tôi muốn nói một chút về những nguyên tắc hoạt động của trò "Người treo cổ". Bây giờ tôi sẽ đưa cho bạn một số chỉ dẫn, đồng thời tôi nghĩ sau này bạn cũng nên giải thích cho người khác biết cách hoạt động của trò chời mà bạn tạo ra.

Đa phần chúng ta đều đã từng biết qua trò "người treo cổ" rồi đúng ko? Nhưng mà bây giờ nhắc lại một chút về cách chơi cũng chả mất gì đúng ko: Mục tiêu của trò chơi là tìm ra từ bị ẩn sau tối đa 10 lần đoán (bạn cũng có thể tự mình thay đổi số lần đoán tối đa để tùy chỉnh độ khó).

Quy cách chơi.

Giả sử từ khóa bị ẩn là RED.

Bạn đoán chữ A và máy tính sẽ kiểm tra xem chữ A có nằm trong từ đang bị ẩn ko.

 Hãy nhớ là có sẵn một hàm trong thư viện *string.h* có thể tìm một chữ cái trong một từ ! (hàm strchr). Tuy nhiên, bạn không cần sử dụng nó (bản thân tôi cũng ít khi dùng đến).

Có 2 khả năng sẽ xảy ra:

- Chữ cái bạn đoán có chứa trong từ bị ẩn, lúc này màn hình sẽ hiển thị chữ mà bạn tìm thấy trong từ bị ẩn đó.
- Chữ cái bạn đoán không có trong từ đang bị ẩn (chẳng hạn như khi nãy bạn đoán chữ A nhưng từ bị ẩn là RED thì làm gì có chữ A): Máy tính sẽ thông báo cho người chơi là chữ vừa được đoán không có chứa trong từ đang bị ẩn và tự động trừ bớt đi một lần đoán của họ. Cho tới khi chúng ta sử dụng hết tất cả các lượt đoán mà vẫn chưa tìm ra từ bị ẩn thì xác định là thua cmnr!!! Đến đây thì GAME OVER.

 Bạn cũng biết trong thực tế khi chơi trò này trên máy tính hoặc kim từ điển thì mỗi lần chúng ta đoán sai, hình ảnh một người bị treo cổ sẽ được vẽ thêm vào 1 nét cho đến khi bạn đoán đúng từ bí ẩn hoặc đoán sai hết thì hình vẽ hoàn thành (cũng có nghĩa là bị treo cổ và thua). Với console thì chúng ta ko thiết kế sinh động như vậy được và chỉ hiển thị chữ cái được thôi, vì vậy có gì xài nấy, chúng ta sẽ hiển thị một câu thông báo "Bạn còn xxx lần đoán trước khi người này bị treo cổ" mỗi khi đoán sai cho đến khi hết lượt đoán.

Giả sử người chơi đoán chữ D (trong trường hợp từ bị ẩn là RED). Rõ ràng là chữ D được chứa trong từ RED, vì vậy số lượt đoán của người chơi sẽ không bị giảm đi. Màn hình sẽ hiển thị từ bị ẩn cùng với chữ cái mà bạn vừa đoán đúng, nó sẽ trông như thế này:

Console:

```
Tu bi an: **D
```

Và nếu sau đó người ta đoán chữ R, lại tiếp tục có chữ R, màn hình sẽ hiển thị một lần nữa từ bí ẩn kèm theo những từ bạn đã đoán đúng, như sau:

Console:

```
Tu bi an: R*D
```

Trường hợp từ chứa nhiều ký tự giống nhau?

Trong một số từ, sẽ có thể xuất hiện 2 hoặc nhiều chữ cái giống nhau. VD: có 2 chữ R trong từ PROGRAM hoặc có 3 chữ E trong EXCELLENT.

Vậy điều gì sẽ xảy ra trong những trường hợp trên? Luật của trò này là: nếu người chơi đoán đúng chữ E trong từ EXCELLENT thì màn hình sẽ hiển thị cả 3 chữ cùng một lúc:

Console:

```
Tu bi an: E**E**E**
```

Người chơi sẽ không cần phải gõ chữ E 3 lần.

Xem thử trò "người treo cổ" hoàn chỉnh.

Sau đây là những gì nên được tạo ra và bạn sẽ thấy chúng xuất hiện trong trò chơi mà bạn sẽ viết:

Console:

```
Chao mung ban tham gia tro choi NGUOI TREO CO!

Ban co 10 luot doan.
Tu bi an la gi?  ********
Hay doan mot chu cai: Z

Ban co 9 luot doan.
Tu bi an la gi?  ********
Hay doan mot chu cai: A

Ban co 9 luot doan.
Tu bi an la gi?  *A******
Hay doan mot chu cai: K

Ban co 9 luot doan.
Tu bi an la gi? *A*****K
Hay doan mot chu cai:
```

Và cứ như vậy cho tới khi người chơi đã tìm được từ bí ẩn (hoặc vượt quá số lượt đoán cho phép):

Console:

```
Ban co 3 luot doan.
Tu bi an la gi? FACEB**K
Hay doan mot chu cai: O

Xin chuc mung! Tu bi mat la:  FACEBOOK
```

Nhập chữ cái vào màn hình console.

Đọc một chữ cái trên màn hình console có lẽ là phức tạp hơn bạn thấy.

Bằng chút trực giác, để có thể nhập một chữ cái bạn nên nghĩ về:

C code:

```
scanf ("%c", &chuCaiBiMat);
```

Và thực tế thì đây là một ý tưởng tốt, %c sẽ được thay bằng một chữ cái, chúng ta sẽ lưu trữ trong biến *chuCaiBiMat* (tất nhiên đây phải là một biến kiểu char).

Tất cả mọi việc đang diễn ra rất tốt … miễn là chúng ta không thay đổi *scanf*. Bạn có thể thử chạy đoạn mã sau:

C code:

```
int main (int argc, char* argv[ ])
{
    char chuCaiBiMat = 0;

    scanf ("%c", &chuCaiBiMat);
    printf ("%c", chuCaiBiMat);

    scanf ("%c", &chuCaiBiMat);
    printf ("%c", chuCaiBiMat);

    return 0;
}
```

Thường thì theo những gì chúng ta đã được học bạn sẽ hiểu đoạn code trên yêu cầu người dùng nhập vào một chữ cái 2 lần sau đó in chúng ra 2 lần.

Ok bạn chạy thử nó đi! Oh, chuyện gì đang xảy ra vậy? Bạn nhập một chữ cái, ok, nhưng … chương trình ngừng ngay sau khi bạn mới nhập chữ cái một lần, còn một lần nhập nữa cơ mà, chương trình đã không yêu cầu bạn nhập tiếp chữ cái tiếp theo. Tại sao vậy, sao nó lại bỏ qua chữ cái thứ 2 trong khi đoạn code trên rõ ràng dùng hàm *scanf* 2 lần để yêu cầu nhập chữ cái 2 lần cơ mà.

 Chuyện gì đã xảy ra?

Thật ra thì, khi bạn nhập một chữ cái ở màn hình console, tất cả mọi thứ bạn gõ vào đã được lưu trữ ở đâu đó trong bộ nhớ, và nó cũng có bao gồm cả ký tự ENTER (\n).

Vì vậy khi bạn nhập ký tự đầu tiên (VD: ký tự A) và bạn bấn ENTER, ký tự A này được trả về bởi hàm *scanf* đầu tiên, sau đó *scanf* tiếp tục trả về ký tự đặc biệt \n tương ứng với phím ENTER mà bạn đã bấm.

Để tránh tình trạng này, tốt hơn hết là hãy tạo một hàm (function) của chúng ta *docKytu( )*:

C code:

```
char docKytu ( )
{
   char kytuNhapVao = 0;

   kytuNhapVao = getchar ( ); // Doc ky tu duoc nhap dau tien
   kytuNhapVao = toupper (kytuNhapVao); // Viet hoa ky tu do

   // Lan luot doc tiep cac ky tu khac cho den khi gap \n
   while (getchar ( ) != '\n') ;

   return kytuNhapVao; // Tra ve ky tu dau tien doc duoc
}
```

Hàm này sử dụng *getchar ( )* đây là một hàm cơ bản có chứa trong thư viện *stdio* cũng như *scanf ("%c" &kytu);*

Hàm *getchar* sẽ đọc dữ liệu được nhập vào, chỉ một ký tự tại một thời điểm từ bàn phím. Khi sử dụng hàm này, các ký tự nằm trong vùng đệm cho đến khi người dùng nhấn phím xuống dòng. Vì vậy nó sẽ đợi cho đến khi phím Enter được gõ. Hàm *getchar( )* không có tham số, nhưng vẫn phải có các cặp dấu ngoặc đơn. Nó đơn giản là lấy về ký tự tiếp theo và sẵn sàng đưa ra cho chương trình. Chúng ta nói rằng hàm này trả về một giá trị có kiểu char.

Sau đó, chúng ta sử dụng một hàm tiêu chuẩn mà bạn chưa có cơ hội được học ở những bài trước: hàm *toupper ( )* giúp viết hoa ký tự. Bằng cách đó, trò chơi vẫn sẽ hoạt động dù cho người chơi có nhập vào một ký tự không viết hoa. Chúng ta sẽ phải khai báo thư viện *#include ctype.h* ở đầu để có thể sử dụng hàm này (đừng quên nhé).

Và bây giờ là phần thú vị nhất: Chúng ta sẽ xóa đi tất cả những ký tự mà người dùng đã nhập để làm trống bộ nhớ. Việc gọi hàm *getchar* sẽ lấy một ký tự tiếp theo như tôi đã giải thích cho bạn ở trên. Giả sử khi người dùng nhấn Enter thì ký tự mà hàm lấy vào sẽ là \n.

Những gì chúng ta phải làm vô cùng đơn giản: chỉ cần một dòng, chúng ta gọi hàm *getchar* trong vòng lặp cho tới khi gặp ký tự \n thì ngừng vòng lặp. Sau khi vòng lặp kết thúc, nó đã đọc tới ký tự \n khi người dùng nhấn Enter, điều đó cũng có nghĩa là hàm *getchar* đã đọc hết những ký tự được người dùng nhập vào trước đó, nhờ đó vùng đệm đã được làm trống theo đúng phương thức hoạt động của hàm *getchar* mà tôi đã nói với bạn ở trên.

 Sao lại xuất hiện dấu chấm phẩy ở vị trí kết thúc dòng chứa vòng lặp while và chúng ta cũng không thấy cặp ngoặc nhọn của vòng lặp như ta đã học ở các bài trước.

Như bạn đã thấy, vòng lặp while của chúng không chứa những câu lệnh (instructions), nó chỉ chứa duy nhất hàm *getchar* trong phần điều kiện. Trong trường hợp này, sử dụng cặp ngoặc nhọn cho vòng lặp là không cần thiết, vì vậy tôi đã sử dụng dấu chấm phẩy để thay cho cặp dấu { }. Dấu chấm phẩy này cũng có nghĩa là "Không cần làm gì mỗi lần qua vòng lặp".

Đây là một chút kỹ thuật đặc biệt được sử dụng bởi rất nhiều lập trình viên và tôi nghĩ bạn cần phải biết để ứng dụng cho những vòng lặp cực ngắn và đơn giản.

Nhưng nếu lỡ bạn không biết kỹ thuật đặc biệt trên thì vòng lặp trong code của bạn sẽ như sau:

C code:

```
while (getchar( ) != '\n')
{

}
```

Không có gì trong cặp dấu ngoặc nhọn là hoàn toàn bình thường, bởi vì trong trường hợp này chúng ta thật sự không có câu lệnh nào cần thực hiện cả. Việc sử dụng dấu chấm phẩy thay thế cho cặp dấu ngoặc nhọn chỉ giúp code của bạn gọn gàng hơn thôi.

Cuối cùng, hàm *docKytu* sẽ trả về ký tự đầu tiên nó đọc được, cũng chính là giá trị của biến *kytu*.

Tóm lại, để có được một từ trong code của bạn, chúng ta sẽ không sử dụng:

C code:

```
scanf ("%c", &chuCaiBiMat);
```

Thay vào đó chúng ta sẽ dùng một cách cao cấp hơn:

C code:

```
chuCaiBiMat = docKytu( );
```

Danh mục từ bí ẩn:

Bắt đầu chương trình test của bạn, tôi sẽ yêu cầu bạn đặt những từ bí mật trực tiếp vào những dòng code. Ví dụ như sau:

C code:

```
char tuBimat [ ] = "LOVE"
```

Nếu chúng ta làm như trên, dĩ nhiên là tất cả các từ bí mật của người chơi sẽ giống như nhau, và chả có gì vui cả đúng không. Tôi yêu cầu bạn làm vậy lúc đầu chỉ để giúp bạn tránh gặp phải những vấn đề rắc rối thôi (chỉ vào thời điểm đó thôi nhé). Sau khi bạn chắc rằng trò chơi đã hoạt động đúng như ý bạn muốn, chúng ta đã có thể thoải mái cải tiến nó sang giai đoạn hai: chúng ta sẽ tạo ra cả một danh mục đầy những từ bí mật khác nhau.

 Cái gì gọi là "danh mục các từ bí mật"?

Nó chính xác là một tập tin chứa rất nhiều từ bí mật cho trò "Người treo cổ" của bạn. Và chắc chắn là mỗi từ sẽ nằm trên 1 dòng. VD như sau:

```
LOVE
MONEY
PROGRAM
FUNCTION
POINTER
LOOP
INSTRUCTION
STRING
VARIABLE
CONSTANT
DEFINE
```

Mỗi lần trò chơi bắt đầu, chương trình của bạn sẽ mở tập tin này lên và chọn ngẫu nhiên một từ trong danh sách. Bằng cách này, bạn sẽ có một tập tin riêng biệt để có thể chỉnh sửa theo ý bạn muốn và thêm vào những từ bí mật mới cho trò "Người treo cổ" thêm thú vị.

 Có thể bạn đã nhận ra rằng từ đầu đến giờ những từ bí mật trong trò chơi đều được tôi viết hoa toàn bộ. Chúng ta sẽ không phân biệt giữa những ký tự viết hoa và viết thường, vậy nên tốt nhất là nên nói rõ ngay từ đầu "tất cả các từ sẽ được viết hoa". Bạn có thể thông báo trước cho người chơi ở phần hướng dẫn, yêu cầu họ nhập ký tự viết hoa chứ không phải chữ thường.\

Hơn nữa, chúng ta cũng bỏ qua những trường hợp từ có dấu để đơn giản hóa trò chơi (nếu chương trình test của chúng ta có những từ có dấu như vậy thì tới giờ này nó vẫn chưa xong đâu). Bạn sẽ cần phải viết những từ bí mật vào tập tin danh mục từ bí mật và chúng phải được viết hoa không dấu.

Có một vấn đề là máy tính sẽ hỏi bạn, có bao nhiêu từ bí mật trong danh mục. Thật vậy, nếu bạn muốn chọn ngẫu nhiên một từ, sẽ có rất nhiều lựa chọn giữa từ mang số thứ tự 0 với một số thứ tự ngẫu nhiên bất kỳ, và máy tính làm sao biết số bất kỳ đó là bao nhiêu.

Để giải quyết vấn đề đó, có hai giải pháp. Bạn có thể định nghĩa số từ chứa trong danh mục ngay dòng đầu tiên của tập tin:

```
3
LOVE
MONEY
LIFE
```

Tuy nhiên, cách này hơi nhàm chán, bởi vì chúng ta sẽ phải tự mình cập nhật lại thông số mỗi lần thêm vào 1 từ trong danh mục (cách này vẫn còn 1 chút bất tiện ở điểm này). Vì vậy tôi gợi ý các bạn có thể tạo ra chức năng tự động đếm số từ có chứa trong tập tin danh mục từ bí ẩn cho chương trình của bạn. Cũng đơn giản thôi, ý tưởng liên quan đến việc đếm ký tự \n mỗi dòng trong tập tin.

Mỗi lần chương trình đọc tập tin, nó sẽ đếm ký tự \n lại từ đầu. Sau đó bạn sẽ lấy kết quả đếm được đó thế vào con số mà chương trình sẽ chọn ngẫu nhiên giữa số 0 và nó để chọn ra từ sẽ được lưu vào bộ nhớ khi trò chơi bắt đầu.

Tôi nhường cho bạn suy nghĩ về giải pháp này đấy. Tôi sẽ không giúp bạn thêm nữa nếu không đây đâu còn là chương trình test của bạn nữa. Hãy sử dụng tất cả những kiến thức mà bạn đã được học, bạn hoàn toàn có thể thiết kế trò chơi này theo như tôi đã nói. Có thể sẽ mất ít nhiều thời gian cũng như bạn sẽ cảm thấy dễ hoặc khó nhưng hãy cố gắng tự mình sắp xếp, tổ chức mọi thứ theo cách của bạn (nhớ tạo đủ function cho những gì bạn muốn chương trình làm nhé), rồi bạn cũng sẽ làm được thôi.

Chúc may mắn, và hơn nữa là "Đừng nản chí" !!!

Giải pháp thứ 1: Mã số trò chơi

Khi bạn đang đọc những dòng này, có thể là bạn đã hoàn tất chương trình hoặc cũng có thể là không.

Bản thân tôi đã mất nhiều thời gian hơn mình nghĩ để biến trò chơi này trông có vẻ khá ngu ngốc. Nó thường là như vậy: nhìn qua thì ta thường nghĩ "ui xời, dễ ẹt" nhưng thực tế thì có một số trường hợp khá phức tạp cần phải giải quyết.

Nhưng tôi đã nói rồi, bạn vẫn có thể tự mình giải quyết được hết, chỉ là mất ít nhiều thời gian hơn thôi. Đây đâu phải là một cuộc chạy đua, bạn cứ thoải mái ra, dành thêm chút thời gian suy nghĩ trước khi tìm đến phần giải pháp này, thử suy nghĩ thêm 5 phút nữa xem sao, biết đâu lại nảy ra được ý tưởng gì đó.

Đừng có nghĩ rằng tôi chỉ việc đặt đít xuống một chút là đã viết xong chương trình. Tôi cũng như các bạn thôi, cũng phải bắt đầu từ những gì đơn giản nhất và cải tiến chúng từ từ để có được kết quả cuối cùng như mình muốn.

Tôi cũng đã từng quên những kiến thức cơ bản như: quên khởi tạo giá trị khi khai báo biến, quên đặt các nguyên mẫu hàm (prototype) hoặc xóa đi những biến không còn giá trị sử dụng trong chương trình để giải phóng bộ nhớ. Thậm chí, tôi phải thú nhận với bạn là tôi còn quên cả việc đặt dấu chấm phẩy ở cuối câu lệnh (instruction).

Tôi nói với bạn những điều trên để làm gì? Tôi không hoàn hảo, và cuộc sống cũng như vậy, luôn có những sai lầm. "LẬP TRÌNH CŨNG NHƯ VẬY … BẠN SẼ TIẾP TỤC VỚI NÓ CHỨ!!! YES OR NO ???"

Ok, bây giờ tôi sẽ cho bạn thấy giải pháp qua 2 giai đoạn:

1. Đầu tiên tôi sẽ cho các bạn thấy làm thế nào để chương trình xử lý các ký tự được ẩn sau mỗi lượt đoán của người dùng. Tôi chọn từ FACEBOOK vì nó cho phép tôi kiểm tra xem mình có giải quyết được trường hợp có 2 ký tự giống nhau trong một từ không.
2. Sau đó tôi sẽ cho bạn thấy làm thế nào để quản lý tập tin danh mục từ bí ẩn để thêm vào các từ bí ẩn cho trò chơi.

Dù có thể nhưng tôi sẽ không viết toàn bộ code cho bạn thấy một lần. Sẽ rất dài và nhìn khá kinh khủng, tôi sợ sẽ làm các bạn bị choáng.

Tôi sẽ cố gắng giải thích lý do tại sao tôi không làm vậy. Hãy khoan quá khao khát đạt được kết quả bạn muốn, mà quan trọng là cách tư duy của chúng ta về vấn đề đó. Hãy tập trung phân tích vào cách giải quyết vấn đề trước nhé.

Phân tích hàm main:

Như mọi người vẫn thường nói rằng "mọi thứ đều bắt đầu từ đôi tay". Hehe. Đừng quên thêm vào những thư viện *stdio*, *stdlib*, *ctype* (thư viện này để sử dụng hàm *toupper ( )* nhé). Chúng thật sự hữu ích và cần thiết cho chương trình của bạn đấy:

C code:

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

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

   return 0;
}
```

Ok, đến lúc này thì chúng ta nên làm theo tôi. Rồi chúng ta sẽ thấy được sự hữu ích của những thư viện, chúng ta sẽ có thể quản lý chương trình và hầu hết các function trong chương trình sẽ cần đến những thư viện này.

Nào, hãy bắt đầu bằng việc khai báo những biến cần thiết. Hãy yên tâm, tôi đã không nghĩ hết về tất cả những biến cần khai báo cho chương trình cùng một lúc, ít ra thì vẫn ít hơn lần đầu tiên tôi tập tành viết code:

C code:

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

int main(int argc, char* argv[ ])
{
   char kytu = 0; // Bien nay se luu tru ky tu cua nguoi choi (duoc tra ve boi ham scanf)
   char tuBimat[ ] = "FACEBOOK"; // Day la tu bi mat can tim ra
   int sokytuBimat[8] = {0}; /* Mot mang co chua cac thanh phan dang Bolean. Moi o trong mang se tuong ung voi mot ky tu nguoi choi se doan. Neu doan dung thi gia tri = 1, va neu sai thi gia tri = 0 */
   int soluotDoan = 10; // So luot doan con lai cua nguoi choi (0 = thua)
   int i = 0; // Bien ho tro

   return 0;
}
```

Tôi đã cố tình khai báo mỗi biến trên một dòng riêng biệt và ghi chú thêm cho từng biến để các bạn dễ hiểu. Trong thực tế, bạn không bắt buộc phải ghi chú tất cả ra như thế và có thể khai báo tất cả những biến cùng kiểu trên cùng một dòng.

Các biến trên được khai báo khá hợp lý đúng không: biến *kytu* để lưu trữ ký tự người chơi đoán được, biến *tuBimat* để lưu trữ từ bí mật của chúng ta, boeén *soluotDoan* để đếm số lượt đoán còn lại của người chơi (lượt đoán = 0 nghĩa là thua) …

Biến *i* là biến hỗ trợ tôi làm việc với mảng, vòng lặp, nó sẽ giúp bạn kết thúc vòng lặp.

Cuối cùng, biến mà bạn cần phải nghĩ đến nhất, thứ khác biệt với những biến còn lại, mảng *sokytuBimat* chứa các thành phần kiểu Bolean. Để ý thì bạn sẽ thấy tôi đã quy định kích thước sẵn cho nó đúng với số ký tự của từ bí ẩn (là 8 ký tự). Không phải một số ngẫu nhiên nhé: mỗi một ô thành phần trong mảng đại diện cho một ký tự của từ bí mật. Ô đầu tiên đại diện cho ký tự đầu tiên, ô thứ 2 là ký tự thứ 2 và cứ thế cho đến ký tự cuối.

Giá trị của các thành phần trong mảng được khởi tạo lúc đầu bằng 0, điều này cũng có nghĩa là "không tìm thấy ký tự nào". Và theo tiến trình hoạt động của game, giá trị này sẽ được thay đổi tương ứng với diễn biến trò chơi. Mỗi một khi có ký tự được tìm đúng, giá trị của ô thành phần trong mảng *sokytuBimat* sẽ mang giá trị 1, ngược lại là giá trị 0.

Ví dụ, nếu đến một lúc người chơi tìm được đến *A*E*OO* thì lúc này mảng của chúng ta sẽ có các giá trị thành phần là 01010110 (giá trị 1 cho mỗi ký tự tìm đúng).

Khá dễ để biết khi nào thì chúng ta thắng trò chơi đúng không. Đó là lúc các giá trị Bolean chỉ toàn là giá trị 1. Ngược lại nếu đoán sai thì giá trị sẽ là 0.

Ok, tiếp tục thôi:

C code:

```
printf ("Chao mung den voi tro choi Nguoi treo co !\n\n ");
```

Đơn giản chỉ là một lời chào mừng thôi mà. Hihihi. Nhưng vòng lặp sau đây mới là điều thú vị:

C code:

```
while (soluotDoan > 0 && !win (sokytuBimat))
{
```

Trò chơi sẽ tiếp tục nếu *soluotDoan* vẫn lớn hơn 0 hoặc chưa thắng. Nếu không còn lượt đoán nào hoặc chiến thắng trò chơi, trong cả 2 trường hợp này, ngừng trò chơi ngay lập tức, và vòng lặp cũng sẽ được kết thúc.

win là một function được phân tích dựa vào mảng *sokytuBimat*. Nó trả về giá trị "true" tương ứng với giá trị 1 nếu người chơi chiến thắng (mảng *sokytuBimat* chỉ chứa toàn giá trị 1), "false" tương ứng giá trị 0 nếu người chơi thua.

Tôi không thể giải thích chi tiết cho bạn về function *win* ngay bây giờ nhưng chúng ta sẽ nói về nó sau ít phút nữa. Bây giờ bạn tạm thời chỉ cần biết chức năng mà nó sẽ làm trong chương trình.

Xem thử nhé:

C code:

```
printf("\n\n Ban co %d luot doan de choi", soluotDoan);
printf("\n Tu bi mat la gi ?");

    /* Hien thi nhung ky tu bi mat va an di nhung ky tu chua duoc tim thay
    Vi du: *A***OO*/
    for (i = 0 ; i < 6 ; i++)
    {
       if (sokytuBimat[i]) // Neu nguoi choi tim duoc ky tu thu i
          printf("%c", tuBimat[i]); // Hien thi ky tu thu i duoc tim thay
       else
          printf("*"); // Hien thi dau * doi voi nhung ky tu chua duoc tim thay
    }
```

Phân tích đoạn code trên, chương trình sẽ hiển thị từng ký tự tìm được và phần còn lại của từ bí mật (phần này được ẩn đi bởi những dấu *). Mỗi khi vòng lặp kết thúc, màn hình sẽ hiển thị những ký tự chưa được tìm thấy bằng dấu * và đồng thời phân tích xem ký tự được người chơi đoán có chứa trong mảng *sokytuBimat* hay không (dựa vào (*if sokytuBimat* [i])).Nếu ký tự được tìm thấy có chứa trong mảng đó thì hiển thị lên, ngược lại thì hiển thị dấu * để ẩn đi.

Bây giờ chúng ta đã có những thứ cần hiển thị rồi, hãy yêu cầu người chơi nhập ký tự họ đoán thôi:

C code:

```
printf ("\n Xin moi ban doan mot ky tu: ");
kytu = docKytu( );
```

Tôi đã gọi hàm *docKytu ( )* của chúng ta. Nó đọc các ký tự đầu tiên được nhập vào, viết hoa nó lên và làm trống bộ nhớ đệm.

C code:

```
// Neu ky tu nhap vao khong dung
if (!kiemtraKytu (kytu, tuBimat, sokytuBimat))
   {
      soluotDoan--; // Giam bot mot lan doan cua nguoi choi
   }
}
```

Chúng ta kiểm tra xem ký tự người chơi nhập vào có chứa trong từ bí mật không. Đoạn code trên gọi hàm *kiemtraKytu* để thực hiện điều đó và tí nữa bạn sẽ thấy cụ thể hàm đó trông như thế nào.

Tạm thời bây giờ những gì chúng ta cần biết là hàm này trả về giá trị "true" nếu ký tự được nhập vào có chứa trong từ bí mật, ngược lại thì trả về giá trị "flase".

Các bạn có để ý *if* của chúng ta bắt đầu bằng một dấu chấm than, điều này có nghĩa là "không". Vậy bây giờ điều kiện của chúng ta sẽ được hiểu là "Nếu không tìm thấy ký tự".

Và điều gì sẽ xảy ra? Nó sẽ giảm bớt một lượt đoán của người chơi.

 Lưu ý rằng hàm *kiemtraKytu* cũng sẽ đặt giá trị vào các ô trong mảng *sokytuBimat*. Nó sẽ đặt giá trị 1 vào những ô nào chứa ký tự đúng được tìm thấy.

Vòng lặp chính của trò chơi cần phải ngừng lại. Vì vậy chúng ta sẽ quay lại đầu vòng lặp và kiểm tra xem người chơi còn lượt đoán nào không và liệu rằng họ đã thắng hay chưa.

Khi ra khỏi vòng lặp chính của trò chơi, chương trình sẽ kiểm tra xem liệu người chơi đã thắng hay chưa trước khi chương trình ngừng lại.

C code:

```
if (win(sokytuBimat))
   printf ("\n\n Chuc mung, ban da chien thang ! Tu bi mat la : %s", tuBimat);
else
   printf ("\n\n Xin chia buon, ban da thua !\n\n Tu bi mat la : %s", tuBimat);

return 0;
}
```

Chúng ta dùng hàm *win* để kiểm tra xem người chơi có thắng hay không để hiển thị thông báo cho họ biết.

Phân tích hàm win:

Bây giờ chúng ta sẽ thấy code của hàm *win*:

C code:

```
int win(int sokytuBimat[ ])
{
   int i = 0;
   int nguoichoiChienThang = 1;

   for (i = 0 ; i < 6 ; i++)
   {
      if (sokytuBimat[i] == 0)
         nguoichoiChienThang = 0;
   }

   return nguoichoiChienThang;
}
```

Hàm này lấy tham số là mảng *sokytuBimat* chứa giá trị dạng Boolean. Hàm sẽ trả về một giá trị "true" nếu người chơi thắng, hoặc giá trị "flase" nếu không thắng.

Code của hàm này nhìn khá đơn giản đúng không, chắc là các bạn đều đọc hiểu đúng không. Chúng ta sẽ kiểm tra thử xem trong mảng *sokytuBimat* có ô nào chứa giá trị 0 không. Nếu xuất hiện bất kỳ ô nào trong mảng có giá trị 0, điều đó có nghĩa là người chơi không thắng được, lúc này biến *nguoichoiChienThang* (biến kiểu Boolean) sẽ mang giá trị "false" tương đương bằng 0. Và nếu tất cả các ký tự đều được tìm thấy, biến này sẽ có giá trị "true" tương đương bằng 1, và hiển nhiên hàm này cũng sẽ trả về giá trị 1 luôn.

Phân tích hàm *kiemtraKytu*:

Hàm *kiemtraKytu* có 2 nhiệm vụ chính:

- Trả về một Boolean chỉ ra rằng ký tự do người dùng đoán có chứa trong từ bí mật hay không.
- Gửi giá trị (vd: giá trị 1) vào các ô giá trị của mảng *sokytuBimat* tương ứng với vị trí của ký tự đó trong mảng.

C code:

```
int kiemtraKytu(char kytu, char tuBimat[ ], int sokytuBimat[ ])
{
   int i = 0;
   int kytuChinhXac = 0;

   // Kiem tra xem ky tu cua nguoi choi da doan co nam trong tu bi mat ko
   for (i = 0 ; tuBimat[i] != '\0' ; i++)
   {
      if (kytu == tuBimat[i]) // Neu ky tu co chua trong tu bi mat
      {
         kytuChinhXac = 1; // Ky tu se duoc luu tru gia tri the hien no la ky tu chinh xac
         sokytuBimat[i] = 1; // Gui gia tri 1 vao o tuong ung voi vi tri cua ky tu do trong mang
      }
   }

   return kytuChinhXac;
}
```

Hàm sẽ kiểm tra ký tự mà người dùng đã nhập vào có nằm trong ký tự bí mật không. Nếu có, có 2 điều sẽ diễn ra:

- Giá trị Boolean của biến *kytuChinhXac* sẽ bằng 1, và giá trị 1 sẽ được gửi vào vị trí tương ứng của ký tự đó trong *kytuBimat*.
- Chúng ta cập nhật giá trị trong mảng *sokytuBimat* tương ứng với vị trí của ký tự đó trong mảng.

Lợi ích của kỹ thuật này là chúng ta sẽ kiểm tra một lượt toàn bộ các giá trị trong mảng (chứ không ngừng lại ngay sau khi tìm được ký tự đầu tiên). Việc này cho phép chúng ta cập nhật chính xác các giá trị trong mảng *sokytuBimat*, kể cả trong trường hợp có 2 ký tự giống nhau trong một từ như 2 ký tự O trong từ FACEBOOK.

Giải pháp 2: Quản lý tập tin "danh mục từ bí mật"

Chúng ta đã có biết tạo ra hầu hết các chức năng cơ bản của trò chơi, và các bạn đã có thể quản lý chương trình của mình nhưng vẫn còn một phần chưa hoàn thiện, đó là cách chọn ngẫu nhiên một từ bí mật trong "danh mục từ bí mật". Bạn có thể tưởng tượng danh mục đó nó giống như tôi đã cho bạn xem ở trên, tôi không cho bạn thấy cụ thể toàn bộ nội dung tập tin đó ở đây được vì có thể nó dài tới vài trang giấy chứ không ít đâu.

Trước khi tiếp tục, việc đầu tiên cần làm là tạo ra một "danh mục từ bí mật" cho trò chơi. Ở thời điểm này. dù nó ngắn hay dài không quan trọng, chỉ là để làm thử nghiệm cho bài học này thôi.

Tôi sẽ tạo một tập tin *danhmuc.txt* trong thư mục chứa project của tôi. Và tạm thời nội dung tập tin sẽ trông như sau:

```
LOVE
MONEY
PROGRAM
FUNCTION
POINTER
LOOP
INSTRUCTION
STRING
VARIABLE
CONSTANT
DEFINE
```

Sau khi hoàn thành chương trình, dĩ nhiên tôi sẽ có thể quay lại danh mục trên và thêm vào tập tin những từ bí mật khác để tăng thêm sự phong phú cho trò chơi.

Những chuẩn bị cho một tập tin mới.

Riêng từ "danh mục" trong tên của tập tin này cũng đủ làm bạn tưởng tượng ra một danh sách có độ dài gần như vô hạn đúng không. Do đó, tôi sẽ thêm một tập tin mới vào project của mình, nó là tập tin *danhmuc.c* (tập tin này có nhiệm vụ đọc *danhmuc.txt*). Và đối với quá trình này, tôi sẽ tạo một tập tin *danhmuc.h*, tập tin này sẽ chứa những prototype của *danhmuc.c*.

Trong *danhmuc.c* tôi sẽ thêm vào các thư viện cần thiết và dĩ nhiên là có cả *danhmuc.h*. Như thường lệ vẫn là những thư viện chuẩn được ưu tiên trước *stdlib, stdio* bên cạnh đó, chúng ta cần chọn một số ngẫu nhiên từ danh mục, vì vậy ta sẽ phải thêm vào thư viện *time.h* giống như đã từng làm với chương trình test của chương 1 (trò "lớn hơn hay nhỏ hơn", bạn còn nhớ ko). Và thêm nữa, ta sẽ sử dụng hàm *strlen* nên bạn phải thêm vào *string.h* nữa nhé:

Xem thử nào:

C code:

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

#include "danhmuc.h"
```

Hàm chonTu:

Hàm này sẽ cần một tham số. Đó là một con trỏ đến vùng bộ nhớ, nơi mà nó có thể lưu trữ từ ngẫu nhiên trong danh mục vào. Con trỏ này sẽ được cung cấp bởi *main ( ).*

Hàm sẽ trả về một giá trị kiểu *int* và nó có kiểu Boolean: giá trị 1 nếu mọi việc diễn ra tốt đẹp và 0 nếu có bất kỳ lỗi gì xảy ra.

Sau đây là đoạn đầu của hàm:

C code:

```
int chonTu(char *tuDuocChon)
{
   FILE* danhmuc = NULL; // Con tro tap tin de chua cac tap tin cua chung ta
   int soThuTuCuaTu = 0, soThuTuCuaTuDuocChon = 0, i = 0;
   int luuKytu = 0;
```

Tôi tạo một số biến cần thiết cho chương trình của chúng ta. Trong *main ( ),* có thể bạn đã thấy tôi đã không tạo ra tất cả biến cùng một lúc ngay khi mới bắt đầu, có những thứ bạn có thể tạo ra sau nếu bạn nhận thấy bạn cần đến chúng.

Tên của các biến trên cũng cho bạn thấy nhiệm vụ, chức năng của chúng rồi. Chúng ta đã có con trỏ *danhmuc* để đọc tập tin *danhmuc.txt*, những biến tạm thời để lưu trữ ký tự …

Bạn có để ý tôi khai báo kiểu *int* cho biến *luuKytu* để lưu trữ ký tự? Hơi lạ đúng không, bởi vì hàm *fgetc* mà tí nữa tôi sẽ dùng nó trả về một giá trị kiểu *int*, vậy nên tốt nhất là nên lưu trữ kết quả của chúng ta kiểu *int*.

Nào xem thử nhé:

C code:

```
danhmuc = fopen("danhmuc.txt", "r"); // Tap tin duoc mo trong che do "read-only"

// Chung ta kiem tra xem thao tac mo tap tin co thanh cong khong
if (danhmuc == NULL) // Neu ban khong the mo tap tin
{
   printf ("\n Khong the mo danh muc tu bi mat");
   return 0; // Tra ve gia tri 0 cho biet thao tac mo tap tin that bai
   // Sau khi nhan duoc gia tri tra ve cua return, ham ket thuc.
}
```

Không có quá nhiều điều mới mẻ ở đây. Tôi chọn chế độ mở tập tin *danhmuc.txt* là "read-only" (bằng cách chọn "r") và kiểm tra xem thao tác này có thành công hay không bằng cách sử dụng *if* nếu giá trị trả của *danhmuc* là NULL thì rõ ràng việc mở tập tin đã thất bại (có thể chương trình không tìm thấy tập tin *danhmuc.txt* hoặc nó đang được chương trình khác sử dụng). Trong trường hợp này, màn hình sẽ hiển thị thông báo lỗi và giá trị trả về là 0.

Tại sao *return* lại nằm ở vị trí đó. Thật ra thì, *return* có chức năng dừng các hoạt động của hàm lại. Nếu không thể mở được tập tin, hàm sẽ dừng lại tại đó và máy tính cũng sẽ không tiếp tục đọc thêm gì nữa. Nó trả về giá trị 0 để cho biết rằng hàm này không thực hiện được.

Và đây là phần còn lại của hàm, giả sử việc mở tập tin đã thành công:

C code:

```
// Dem cac tu duoc chua trong tap tin (chi viec dem co bao nhieu ky tu \n thoi)
do
{
   luuKytu = fgetc(danhmuc);
   if (luuKytu == '\n')
      soThuTuCuaTu ++;
} while(luuKytu != EOF);
```

Bạn thấy gì không, chúng ta sẽ đọc qua toàn bộ tập tin nhờ vào hàm *fgetc* (và đọc lần lượt từng ký tự một nhé). Chương trình chỉ việc đếm số lần xuất hiện ký tự \n. Mỗi lần ký tự \n xuất hiện, giá trị của biến *soThuTuCuaTu* sẽ tăng thêm 1.

Với phần code này, chúng ta sẽ biết được có bao nhiêu từ bí mật nằm trong tập tin. Và nhớ là mỗi một dòng trong tập tin chỉ chứa 1 từ thôi nhé.

C code:

```
soThuTuCuaTuDuocChon = tuNgauNhien(soThuTuCuaTu); // Chon mot tu ngau nhien
```

Như bạn thấy, tôi gọi một hàm theo kiểu riêng của mình, chọn ra từ có số thứ tự ngẫu nhiên từ 1 đến giá trị của *soThuTuCuaTu* (tham số này sẽ được gửi vào hàm).

Đây là một hàm đơn giản mà tôi đã đặt vào file *danhmuc.c* (tôi sẽ cho bạn thấy rõ chi tiết tí nữa). Nó sẽ trả về một giá trị số (tương ứng với số thứ tự của dòng chứa từ đó trong tập tin), giá trị này sẽ được gửi vào biến *soThuTuCuaTuDuocChon.*

C code:

```
// Chuong trinh doc lai tu dau tap tin va ngung lai khi tim thay tu ngau nhien duoc chon
rewind(danhmuc);
while (soThuTuCuaTuDuocChon > 0)
{
   luuKytu = fgetc(danhmuc);
   if (luuKytu == '\n')
      soThuTuCuaTuDuocChon --;
}
```

Bây giờ thì chúng ta đã có được số thự tự của từ được chọn ngẫu nhiên, bằng việc gọi hàm *rewind ( )* chương trình sẽ bắt đầu đọc tập tin từ đầu. Nó sẽ đếm lần lượt từng ký tự \n trong tập tin. Lần này, giá trị của biến *soThuTuCuaTuDuocChon* sẽ giảm xuống. Vd từ được chọn ngẫu nhiên là từ thứ 5 trong tập tin, thì sau mỗi vòng lặp giá trị này sẽ giảm xuống còn 4, 3, 2, 1 và 0. Khi giá trị của *soThuTuCuaTuDuocChon* bằng 0 thì chúng ta sẽ thoát ra khỏi vòng lặp, vì lúc này nó không còn thỏa mãn điều kiện biến này có giá trị lớn hơn 0 nữa.

Bạn cần phải hiểu được ý nghĩa của đoạn code này, nó giúp ta có thể tìm được vị trí hiện tại của "dấu nháy ảo" trong tập tin. Điều này cũng không thực sự quá khó hiểu nhưng bạn phải cố gắng nhìn nhận mọi việc một cách rõ ràng. Hãy cố hết sức đảm bảo rằng bạn hiểu được những gì tôi đang nói và đang làm.

Bây giờ chúng ta cần một "dấu nháy" để chỉ ra vị trí của từ được chọn trong tập tin.

Chúng ta sẽ gửi đến con trỏ *tuDuocChon* (đây là tham số mà hàm cần) với *fgets* để đọc các từ:

C code:

```
/* Con tro tap tin danhmuc duoc dat dung vi tri cua no.
Chung ta su dung ham fgets va quy dinh ham khong doc qua so luong ky tu cho phep*/
fgets(tuDuocChon, 100, danhmuc);

// Chung ta se thay the ky tu \n
tuDuocChon[strlen(tuDuocChon) - 1] = '\0';
```

Chúng ta yêu cầu hàm *fgets* không được đọc quá 100 ký tự (cũng tương ứng với kích thước của mảng). Hãy nhớ rằng hàm *fgets* sẽ đọc toàn bộ một dòng bao gồm luôn ký tự **\n**.

Khi bạn không muốn gặp ký tự \n này ở cuối, có thể thay thế bằng \0, ký tự này có tác dụng cắt một chuỗi trước khi gặp ký tự \n.

Và … nó đã hoạt động … chúng ta đã lưu trữ ký tự bí mật tại địa chỉ con trỏ *tuDuocChon*.

Chúng ta vẫn chưa đóng tập tin lại, một giá trị trả về 1 để ngừng hàm lại và cho thấy rằng chương trình hoạt động tốt:

C code:

```
fclose(danhmuc);

   return 1; // Gia tri tra ve = 1, tat ca deu hoat dong tot
}
```

Và tôi cũng không có gì để nói thêm về hàm *chonTu.*

Hàm tuNgauNhien:

Đây là hàm mà tôi đã hứa sẽ giải thích cho bạn khi nãy. Nó sẽ chọn một số thứ tự ngẫu nhiên và gửi nó về:

C code:

```
int tuNgauNhien(int sothutuLonNhat)
{
   srand(time(NULL));
   return (rand( ) % sothutuLonNhat);
}
```

Dòng đầu tiên srand(time(NULL)); sẽ giúp chương trình của bạn không chọn trùng các số ngẫu nhiên, giống những gì chúng ta đã từng làm với chương trình "lớn hơn hay nhỏ hơn" trong chương 1.

Dòng thứ 2 sẽ chọn ra một số ngẫu nhiên trong chuỗi giá trị từ 0 đến giá trị của biến *sothutuLonNhat.*

Tập tin danhmuc.h

Sau đây là prototype của các function. Và một điều nữa, bạn có còn nhớ về "người bảo vệ" #ifndef mà tôi đã từng yêu cầu các bạn nên thêm vào tất cả những tập tin .h của bạn (tham khảo lại những kiến thức của các bài học về prototype và tiền xử lý nhé).

C code:

```
#ifndef DEF_DANHMUC
#define DEF_DANHMUC

int chonTu(char *tuDuocChon);
int tuNgauNhien(int sothutuLonNhat);

#endif
```

Tập tin danhmuc.c

Và sau đây là toàn bộ nội dung của tập tin *danhmuc.c*, xin mời các bạn thưởng thức:ư

C code:

```
/*
Nguoi Treo Co

danhmuc.c
------

Nhung function nay se chon ra mot tu ngau nhien trong tap tin chua danh muc tu bi an cua tro choi
Nguoi Treo Co
*/


#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#include "danhmuc.h"


int chonTu(char *tuDuocChon)
{
   FILE* danhmuc = NULL; // Con tro tap tin de chua cac tap tin cua chung ta
   int soThuTuCuaTu = 0, soThuTuCuaTuDuocChon = 0, i = 0;
   int luuKytu = 0;
   danhmuc = fopen("danhmuc.txt", "r"); // Tap tin duoc mo trong che do "read-only"


   // Chung ta kiem tra xem thao tac mo tap tin co thanh cong khong
   if (danhmuc == NULL) // Neu ban khong the mo tap tin
   {
      printf ("\n Khong the mo danh muc tu bi mat");
      return 0; // Tra ve gia tri 0 cho biet thao tac mo tap tin that bai
      // Sau khi nhan duoc gia tri tra ve cua return, ham ket thuc.
   }
```

```
   // Dem cac tu duoc chua trong tap tin (chi viec dem co bao nhieu ky tu \n thoi)
   do
   {
      luuKytu = fgetc(danhmuc);
      if (luuKytu == '\n')
         soThuTuCuaTu++;
   } while(luuKytu != EOF);

   soThuTuCuaTuDuocChon = tuNgauNhien(soThuTuCuaTu); // Chon mot tu ngau nhien

   // Chuong trinh doc lai tu dau tap tin va ngung lai khi tim thay tu ngau nhien duoc chon
   rewind(danhmuc);
   while (soThuTuCuaTuDuocChon > 0)
   {
      luuKytu = fgetc(danhmuc);
      if (luuKytu == '\n')
         soThuTuCuaTuDuocChon--;
   }

   /* Con tro tap tin danhmuc duoc dat dung vi tri cua no.
   Chung ta su dung ham fgets va quy dinh ham khong doc qua so luong ky tu cho phep*/
   fgets(tuDuocChon, 100, danhmuc);

   // Chung ta se thay the ky tu \n
   tuDuocChon[strlen(tuDuocChon) - 1] = '\0';
   fclose(danhmuc);

   return 1; // Gia tri tra ve = 1, tat ca deu hoat dong tot
}

int tuNgauNhien(int sothutuLonNhat)
{
   srand(time(NULL));
   return (rand( ) % sothutuLonNhat);
}
```

Chúng ta sẽ làm vài thứ trong main.c:

Bây giờ tập tin *danhmuc.c* đã sẵn sàng, chúng ta cùng trở lại với *main.c* để thêm vào một số thứ cho phù hợp.

Với những gì đã làm từ đầu đến giờ, trước hết chúng ta sẽ thêm vào file *danhmuc.h* nếu bạn muốn sử dụng các function của *danhmuc.c*. Ngoài ra, chúng ta cũng sẽ không quên khai báo thêm thư viện *string.h* bởi vì chúng ta sẽ phải sử dụng hàm *strlen*:

C code:

```
#include <string.h>
#include "dico.h"
```

Để bắt đầu, việc khai báo các biến sẽ có một chút thay đổi. Bạn không cần khởi tạo một chuỗi ký tự *tuBimat* bởi vì chúng ta đã có sẵn một mảng kiểu char cho nó (chứa sẵn 100 ô luôn).

Với mảng *sokytuBimat*, kích thước của nó phụ thuộc vào độ dài của từ bí mật. Bởi vì chúng ta chưa biết được kích thước đó, nên sẽ có một con trỏ được tạo ra, cùng với *malloc* chúng ta có thể gửi con trỏ đến vị trí bộ nhớ mà nó sẽ được cấp phát.

Đây là một vd tuyệt vời cho bài học về cấp phát động của chúng ta trước đây: chúng ta không thể nào biết trước được kích thước của mảng nếu chương trình chưa chạy, vì vậy bắt buộc bạn phải tạo ra một con trỏ và sử dụng hàm *malloc*.

Bạn không được quên giải phóng bộ nhớ để còn sử dụng cho những mục đích khác nhé. Đó cũng chính là lý do cho sự xuất hiện của *free ( )* ở cuối *main.c*.

Chúng ta sẽ cần một biến *dodaiTu* để lưu trữ giá trị thể hiện số lượng ký tự của từ. Như bạn đã thấy trong phần đầu, chúng ta đã giả định từ bí mật sẽ có 8 ký tự (bởi vì chúng ta chọn FACEBOOK làm vd mà). Nhưng bây giờ chúng ta đã có thể làm việc linh động với mọi từ với kích thước tùy biến.

Và sau đây là đầy đủ tất cả những biến cần cho chương trình của chúng ta:

C code:

```
int main(int argc, char* argv[ ])
{
   char kytu = 0; // Bien nay se luu tru ky tu cua nguoi choi (duoc tra ve boi ham scanf)
   char tuBimat[100] = {0}; // Day la tu bi mat can tim ra
   int *sokytuBimat = NULL; /* Mot mang co chua cac thanh phan dang Bolean. Moi o trong mang se tuong ung voi mot ky tu nguoi choi se doan. Neu doan dung thi gia tri = 1, va neu sai thi gia tri = 0 */
   int soluotDoan = 10; // So luot doan con lai cua nguoi choi (0 = thua)
   int i = 0; // Bien ho tro
   int dodaiTu = 0;
```

Trên đây chủ yếu là những thay đổi ban đầu, bây giờ chúng ta sẽ đi sâu hơn một chút:

C code:

```
if (!chonTu(tuBimat))
   exit(0);
```

Đầu tiên chúng ta đặt hàm *chonTu* vào phần điều kiện của if, hàm này sẽ lấy tham số là biến *tuBimat.*

Và như ta đã biết, hàm sẽ trả về một giá trị Boolean (1 hoặc 0) để cho biết nó có được thực hiện thành công hay không. Vai trò của *if* là phân tích các giá trị Boolean được trả về bởi hàm *chonTu*, nếu hàm KHÔNG hoạt động (hãy lưu ý trong if có chứa dấu ! để thể hiện phủ định), thì ngừng lại tất cả bằng *exit (0).*

C code:

```
dodaiTu = strlen(tuBimat);
```

Giá trị thể hiện kích thước của *tuBimat* được lưu trữ trong biến *dodaiTu.*

C code:

```
sokytuBimat = malloc(dodaiTu * sizeof(int)); /* mang sokytuBimat se duoc cap phat dong bo nho (luc dau chung ta khong biet duoc kich thuoc cua mang nay) */
if (sokytuBimat == NULL)
   exit(0);
```

Bây giờ chúng ta phải cấp phát bộ nhớ cho mảng *sokytuBimat.* Kích thước của mảng này đã được cung cấp bởi giá trị của biến *dodaiTu.*

Sau đó dùng if để kiểm tra giá trị của con trỏ có bằng NULL không. Trong trường hợp nếu bằng NULL, chứng tỏ việc cấp phát bộ nhớ đã thất bại, chúng ta sẽ dừng chương trình ngay lập tức (bằng cách gọi hàm *exit (0)*)

Đây là tất cả những gì bạn cần chuẩn bị cho chương trình. Sâu đây chúng ta cần thay đổi nốt phần còn lại của *main.c* để thay thế tất cả các con số 8 (số thể hiện độ dài của từ FACEBOOK mà ta giả định là từ bí mật lúc đầu bài học) bằng biến *dodaiTu.* VD:

C code:

```
for (i = 0 ; i < dodaiTu ; i++)
   sokytuBimat[i] = 0;
```

Ban đầu, đoạn code trên đặt giá trị 0 vào các ô trong mảng *sokytuBimat* và có thể hiểu là không có ô nào trong mảng, dần dần giá trị đó sẽ tăng lên cho đến khi nào bằng với giá trị của biến *dodaiTu* thì ngừng lại, và đó cũng chính là kích thước chính xác cho mảng.

Tôi cũng phải thiết kế lại prototype của function *win* để thêm vào biến *dodaiTu*. Nếu không các hàm sẽ không biết khi nào phải dừng vòng lặp.

Sau đây là tập tin *main.c* hoàn thiện:

C code:

```
/*
Nguoi Treo Co

main.c
------

Nhung function nay se chon ra mot tu ngau nhien trong tap tin chua danh muc tu bi an cua tro choi Nguoi Treo Co
*/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#include "danhmuc.h"

int win(int sokytuBimat[ ], long dodaiTu);
int kiemtraKytu(char kytu, char tuBimat[ ], int sokytuBimat[ ]);
char docKytu( );

int main(int argc, char* argv[ ])
{
   char kytu = 0; // Bien nay se luu tru ky tu cua nguoi choi (duoc tra ve boi ham scanf)
   char tuBimat[100] = {0}; // Day la tu bi mat can tim ra
   int *sokytuBimat = NULL; /* Mot mang co chua cac thanh phan dang Bolean. Moi o trong mang se tuong ung voi mot ky tu nguoi choi se doan. Neu doan dung thi gia tri = 1, va neu sai thi gia tri = 0 */
   long soluotDoan = 10; // So luot doan con lai cua nguoi choi (0 = thua)
   long i = 0; // Bien ho tro
   long dodaiTu = 0;

   printf ("Chao mung den voi tro choi Nguoi treo co !\n\n ");

   if (!chonTu(tuBimat))
      exit(0);
```

```
   dodaiTu = strlen(tuBimat);

   sokytuBimat = malloc(dodaiTu * sizeof (int)); /* mang sokytuBimat se duoc cap phat dong bo
nho (luc dau chung ta khong biet duoc kich thuoc cua mang nay) */
   if (sokytuBimat == NULL)
      exit(0);

   for (i = 0 ; i < dodaiTu ; i++)
      sokytuBimat[i] = 0;

   /* Chung ta se tiep tuc tro choi neu con it nhat mot luot doan
     Va van chua tim duoc tu bi mat*/
   while (soluotDoan > 0 && !win(sokytuBimat, dodaiTu))
   {
      printf ("\n\n Ban co %d luot doan de choi ", soluotDoan);
      printf ("\n Tu bi mat la gi ?");

      /* Hien thi nhung ky tu bi mat va an di nhung ky tu chua duoc tim thay
      Vi du: *A***OO*/
      for (i = 0 ; i < dodaiTu ; i++)
      {
         if (sokytuBimat[i]) // Neu nguoi choi tim duoc ky tu thu i
            printf ("%c", tuBimat[i]); // Hien thi ky tu thu i duoc tim thay
         else
            printf ("*");// Hien thi dau * doi voi nhung ky tu chua duoc tim thay
      }

      printf ("\n Xin moi ban doan mot ky tu: ");
      kytu = docKytu( );

      // Neu ky tu nhap vao khong dung
      if (!kiemtraKytu(kytu, tuBimat, sokytuBimat))
      {
         soluotDoan--; // Giam bot mot lan doan cua nguoi choi
      }
   }

   if (win(sokytuBimat, dodaiTu))
      printf ("\n\n Chuc mung, ban da chien thang ! Tu bi mat la : %s", tuBimat);
```

```
    else
        printf ("\n\n Xin chia buon, ban da thua !\n\n Tu bi mat la : %s", tuBimat);

    free (sokytuBimat); // Giai phong bo nho da duoc phan bo (boi ham malloc)

        return 0;
}

char docKytu( )
{
    char kytuNhapVao = 0;

    kytuNhapVao = getchar( ); // Doc ky tu duoc nhap dau tien
    kytuNhapVao = toupper(kytuNhapVao); // Viet hoa ky tu do

    // Lan luot doc tiep cac ky tu khac cho den khi gap \n
    while (getchar( ) != '\n') ;

    return kytuNhapVao; // Tra ve ky tu dau tien doc duoc
}

int win(int sokytuBimat[ ], long dodaiTu)
{
    long i = 0;
    int nguoichoiChienThang = 1;

    for (i = 0 ; i < dodaiTu ; i++)
    {
        if (sokytuBimat[i] == 0)
            nguoichoiChienThang = 0;
    }

    return nguoichoiChienThang;
}

int kiemtraKytu(char kytu, char tuBimat[ ], int sokytuBimat[ ])
{
    long i = 0;
    int kytuChinhXac = 0;
```

```
  // Kiem tra xem ky tu cua nguoi choi da doan co nam trong tu bi mat ko
  for (i = 0 ; tuBimat[i] != '\0' ; i++)
  {
     if (kytu == tuBimat[i]) // Neu ky tu co chua trong tu bi mat
     {
        kytuChinhXac = 1; // Ky tu se duoc luu tru gia tri the hien no la ky tu chinh xac
        sokytuBimat[i] = 1; // Gui gia tri 1 vao o tuong ung voi vi tri cua ky tu do trong mang
     }
  }

  return kytuChinhXac;
}
```

Ý tưởng cải tiến:

Chà, trò "Người treo cổ" này có hơi phức tạp với bạn không. Bây giờ bạn đã có một chương trình chọn ra từ ngẫu nhiên từ một tập tin rồi đúng không.

Sau đây là một số ý tưởng cải tiến để bạn thử sức:

- Hiện nay, chúng ta chỉ mới cho phép mọi người chơi một lần. Nghĩa là chương trình sẽ ngừng lại khi có người chiến thắng hoặc sử dụng hết lượt đoán. Bây giờ, bạn hãy thử tạo một yêu cầu người chơi xem họ có muốn chơi lại không.
- Bạn cũng có thể tạo ra chế độ chơi 2 người, người thứ nhất sẽ nhập từ bí ẩn vào cho người thứ 2 đoán.
- Mặc dù không bắt buộc nhưng sao bạn không thử vẽ hình người bị treo cổ trên màn hình console (như các kim từ điển hay có) gợi ý là bạn có thể làm bằng hàm *printf*.

Hãy cố gắng bỏ thời gian ra để hiểu bài học này và cải tiến nó hết mức có thể nhé.

Cố lên nào, đừng nản lòng.
