ĐẠI HỌC HUẾ
Trường Đại học Kinh tế
Khoa Hệ thống Thông tin Kinh tế

Cơ sở lập trình
Bài tập

Hạn cuối nộp bài : ***24h00 thứ bảy, ngày 09 tháng 12 năm 2017***
Nộp bằng cách gởi email về địa chỉ ***lvman@hce.edu.vn***
Hãy đọc kỹ ***Quy định nộp báo cáo bài tập*** *trước khi làm bài*

Giới thiệu

Trong bài tập này, các bạn sẽ vận dụng tất cả các khái niệm, kiến thức đã học trong môn Cơ sở lập trình và sử dụng ngôn ngữ C# để thiết kế, xây dựng game Hangman.

Game Hangman

Hangman, còn gọi là người treo cổ, là một trò chơi đoán chữ dựa vào số ký tự của từ. Thông thường, trò này gồm hai người chơi với phương tiện đơn giản là bút và một tờ giấy trắng. Người thứ nhất sẽ nghĩ một từ trong đầu và đưa ra số ký tự của từ đó dưới dạng một hàng gạch ngang, người thứ hai sẽ lần lượt đoán các chữ cái mà họ cho là có mặt trong từ, mỗi lần đoán đúng thì các gạch ngang tương ứng sẽ được thay bằng chữ cái đoán được, mỗi lần đoán sai thì "giá treo cổ"sẽ được vẽ thêm một nét của "người treo cổ". Cuộc chơi kết thúc khi từ được đoán đúng hoặc hình người treo cổ (gồm 6 nét, tương ứng 6 lần đoán sai) được hoàn tất.

Các bạn sẽ lập trình game Hangman để cho phép người chơi đấu với máy tính. Mỗi lượt chơi, máy tính sẽ chọn ngẫu nhiên ra một từ (trong danh sách từ đọc từ tập tin *words.txt* hoặc tập tin có cấu trúc tương tự do người dùng đưa vào theo đối số dòng lệnh) và thông báo số ký tự của từ đó. Người chơi sẽ đoán ký tự. Máy tính sẽ dựa trên ký tự người chơi nhập vào để thông báo đúng hay sai. Nếu đúng thì ký tự tương ứng sẽ được hiển thị ra. Nếu đoán sai thì hình "người treo cổ"sẽ được vẽ thêm 1 nét.

Nếu người chơi đoán đúng được tất cả các ký tự trước khi đạt đến 6 lần đoán sai thì người chơi thắng. Nếu ngược lại, người chơi thua. Máy tính sẽ cho phép người chơi chơi tiếp ván khác nếu muốn.

Chương trình có thêm các chức năng sau:

- Hiển thị menu chương trình để cho người chơi lựa chọn chức năng

- Hiển thị danh sách những người chơi thắng trong thời gian ngắn nhất
- Lưu tên người chơi và thời gian để thắng 1 ván

Yêu cầu của bài tập

Hãy sử dụng tất cả các kiến thức về ngôn ngữ lập trình C# đã được học để lập trình tạo ra chương trình được mô tả như trên. Chương trình cần có các chức năng sau :

1. Có cấu trúc dữ liệu để lưu trữ: từ được chọn để người chơi đoán, các ký tự đã đoán và số lần đã đoán sai.
2. Đọc danh sách *từ* từ tập tin *words.txt*
3. Lựa chọn 1 từ ngẫu nhiên để chơi
4. Kiểm tra các ký tự đã đoán đã trùng với từ cần đoán hay chưa
5. In chuỗi ký tự đã đoán đúng theo thứ tự của từ cần đoán (ký tự nào chưa đoán được thì dùng dấu gạch ngang)
6. Cho người chơi chơi 1 ván mới
7. Gọi hàm để vẽ "người treo cổ"(sử dụng thư viện thầy cung cấp tại Kho github cho môn CSLT)
8. Menu chương trình cho phép người chơi lựa chọn chức năng (Xem ví dụ tại slide 21-22, chủ đề 5 - Các cấu trúc điều khiển 2)
9. Cho phép người chơi đoán luôn cả từ (trừ 2 lượt đoán nếu đoán sai)
10. In ra những ký tự người chơi chưa đoán được khi người chơi thua
11. (Tuỳ chọn - cộng điểm) Hỗ trợ đối số dòng lệnh để cho phép người chơi đưa vào danh sách từ mới (Xem các slide 37-40, chủ đề 7 - Mảng)
12. (Tuỳ chọn - cộng điểm) Lưu và hiển thị danh sách 10 người chơi thắng trong thời gian ngắn nhất

Chú ý :

1. Nên tổ chức chương trình dưới dạng hàm để dễ kiểm soát và sửa lỗi.
2. Trong mỗi chức năng các bạn nên xây dựng thành các hàm chức năng nhỏ hơn. Như vậy, chương trình sẽ không bị rối và dễ gỡ lỗi. Đồng thời các chức năng khác có thể sử dụng lại các hàm đó.

3. Nên code từng phần một, sau mỗi phần phải kiểm tra kỹ càng đoạn code vừa được viết là hoạt động tốt rồi mới chuyển sang phần khác. *Ví dụ : sau khi code phần cho phép đọc dữ liệu từ file, thì nên viết đoạn code in dữ liệu ra để kiểm soát việc đọc dữ liệu vào là đúng.*

4. Khi kiểm tra thì nên kiểm tra với lượng ít số liệu để dễ kiểm soát chương trình. Sau khi chương trình đã đúng với số ít dữ liệu đó thì mới chạy chương trình với nhiều dữ liệu hơn.

Test

Các bạn dùng tập tin *words.txt* kèm theo Bài tập này để kiểm tra chương trình và chụp ảnh màn hình kết quả để dán vào báo cáo. Yêu cầu chạy test như sau:

1. Cho 5 người chơi, mỗi người chơi 3 ván. Nếu đoán ra được từ thì nhập cả từ vào để hoàn tất sớm việc đoán từ.

2. Chụp ảnh màn hình Console mỗi bước chơi, lưu vào thư mục images, đánh số các ảnh theo thứ tự từng bước chơi. Nén thư mục ảnh này lại và gởi kèm trong email nộp bài tập.

3. Hiển thị danh sách 10 người chơi thắng trong thời gian ngắn nhất

Hướng dẫn lập trình

Những thuật toán, cấu trúc dữ liệu và cách xử lý được đưa ra trong các phần sau đây chỉ là **đề nghị**. *Các bạn có thể đưa ra thuật toán, cấu trúc dữ liệu và cách xử lý riêng và hãy chỉ ra trong báo cáo của bạn.* Thuật toán, cấu trúc dữ liệu và cách xử lý của bạn là tốt hơn sẽ được đánh giá cao hơn và ngược lại, điểm của bạn sẽ thấp nếu chương trình của bạn hỗ trợ ít chức năng hơn hoặc cách xử lý không tốt.

Các phần dưới đây sẽ được trình bày tuần tự theo một quy trình mà nếu các bạn làm theo từng bước thì có thể hoàn thành tốt chương trình. Bước sau sẽ là phát triển tiếp của bước trước, nên không làm bước trước thì sẽ khó hoàn thành được bước sau.

Cấu trúc dữ liệu để lưu trạng thái của chương trình

Cấu trúc dữ liệu là một trong hai thành phần cốt lõi để tạo nên một chương trình máy tính và nó có tính quyết định đến việc lựa chọn thuật toán, cách xử lý trong chương trình. Do đó, khi phân tích thiết kế chương trình, ta nên lựa chọn cấu trúc dữ liệu tốt để việc xử lý sau này sẽ đơn giản và thuận tiện hơn.

Đầu tiên, chúng ta cần phân tích xem chương trình cần lưu những dữ liệu gì. Theo mô tả của bài toán, chúng ta cần lưu trữ các thông tin sau: từ được chọn để người chơi đoán (`secret_word`), các ký tự đã đoán (`letters_guessed`) và số lần đã đoán sai (`mistakes_made`).

Về danh sách các từ trong tập tin words.txt sẽ đề cập đến ở mục Đọc danh sách từ từ tập tin words.txt.

Với thông tin đầu tiên "từ được chọn để người chơi đoán", chúng ta có thể dùng một biến kiểu chuỗi ký tự để lưu trữ. Với thông tin thứ ba "số lần đoán sai", chúng ta có thể dùng một biến kiểu số nguyên để lưu trữ. Còn với thông tin "các ký tự đã đoán", thì chúng ta có thể sử dụng một mảng hoặc `List` để lưu trữ. Mảng rõ ràng là đơn giản nhất, nhưng mảng lại không có tính linh động khi chúng ta cần mở rộng kích thước của mảng, `List` thì linh động hơn. Dù là mảng kém linh động như vậy nhưng chúng ta vẫn có thể dùng. Nếu sử dụng mảng, bạn sẽ cấp phát số thành phần cố định ban đầu cho mảng và khi đó, bạn chỉ có thể lưu trữ nhiều nhất bằng số lượng cố định đó. Do đó, bạn hãy dự đoán số thành phần lớn nhất mà mảng cần có thật tốt.

Đọc danh sách *từ* từ tập tin *words.txt*

Bài tập này cung cấp một từ điển (trong tập tin *words.txt* kèm theo Bài tập) để ứng dụng có thể tải lên và lựa chọn từ. Nội dung của tập tin *words.txt* gồm 55.900 từ được ghi trên **1 dòng**, các từ cách nhau bằng **1 khoảng trống**.

Các bạn sẽ dùng lớp `StreamReader` (Xem Chủ đề 8 - slide 15, 16, 17) để đọc nội dung trong file này. Rồi dùng hàm `Split` của lớp `String` (Xem Chủ đề 7 - slide 35) để tách các từ ra.

Thuật toán đọc dữ liệu từ tập tin từ điển như sau :

Đầu vào : đường dẫn đến tập tin từ điển

Đầu ra : mảng kiểu chuỗi ký tự chứa các từ đọc được

```
    Khởi tạo đối tượng kiểu StreamReader với đường dẫn đến tập tin từ điển
    cần đọc
    Đọc dòng văn bản trong tập tin
    Tách các từ ra khỏi dòng văn bản, gán vào 1 mảng kiểu chuỗi ký tự
    Trả mảng kết quả ra
```

Lựa chọn 1 từ ngẫu nhiên để chơi

Để lựa chọn một từ ngẫu nhiên trong từ điển, chúng ta sẽ dùng đến lớp `Random` để sinh ra một số ngẫu nhiên trong phạm vi từ 1 đến 55.900 (với tập tin words.txt). Các bạn có thể xem thông tin và ví dụ sử dụng lớp Random tại trang MSDN ở đây.

Kiểm tra các ký tự đã đoán trùng với các từ cần đoán hay chưa

Ở đây, chúng ta sẽ viết một hàm kiểm tra xem người chơi đã đoán đúng từ cần đoán hay chưa dựa trên các ký tự đã đoán trong `letters_guessed` và từ cần đoán trong `secret_word`. Ở đây, rõ ràng bạn sẽ dùng 1 vòng lặp. Tuy nhiên, bạn sẽ có thể lặp qua 2 thứ: một là các ký tự trong `secret_word` hay là các ký tự trong `letters_guessed`.

Một thuật toán cho hàm này, lặp qua `secret_word` như sau:

Đầu vào : *secret_word* và *letters_guessed*
Đầu ra : *true* nếu tất cả các ký tự trong *secret_word* đều có trong *letters_guessed*
false nếu ngược lại
Lặp qua từng ký tự trong *secret_word*
Nếu ký tự đó không có trong *letters_guessed*
Trả ra kết quả *false*
Trả ra kết quả *true*

In chuỗi ký tự đã đoán đúng theo thứ tự của từ cần đoán

Ví dụ: Từ cần đoán là *claptrap*.

Nếu người chơi chưa đoán đúng ký tự nào cả, tức là `letters_guessed` là rỗng, chương trình sẽ in ra:

```
--------
```

Nếu `letters_guessed` là ['a','p'], chương trình sẽ in ra:

```
--ap--ap
```

Nếu `letters_guessed` là ['a','l','m','c','e','t','r','p','n'], chương trình sẽ in ra:

```
claptrap
```

Thuật toán để tạo ra chuỗi ký tự trên như sau:

Đầu vào : *secret_word* và *letters_guessed*
Đầu ra : chuỗi ký tự *st*
Khởi tạo chuỗi rỗng cho biến *st*
Lặp qua từng ký tự trong *secret_word*
Nếu ký tự đó có trong *letters_guessed*
Thêm ký tự đó vào cuối chuỗi *st*
Nếu không có
Thêm ký tự − vào cuối chuỗi *st*
Trả *st* ra

Cho người chơi chơi 1 ván mới

Đến đây, các bạn có thể kết hợp tất cả các đoạn chương trình ở trên lại để cho phép người chơi có thể chơi 1 ván.

Thuật toán cho 1 ván chơi như sau:

```
Lấy secret_word
Khởi tạo rỗng cho letters_guessed
Khởi tạo giá trị 0 cho mistakes_made
In chuỗi ký tự cần đoán dưới dạng các dấu gạch ngang
Thông báo số lần còn thế thể dự đoán
Cho người dùng nhập vào ký tự dự đoán
Thêm ký tự đó vào letters_guessed
Kiểm tra xem người dùng đã đoán đúng từ cần đoán hay chưa
    Nếu chưa
        Nếu số lần dự đoán đã đủ 6 lần
            Thông báo người chơi đã thua
        Ngược lại, quay lại bước 4
    Nếu đã đúng, thì thông báo người chơi thắng và in ký tự dự đoán
```

Gọi hàm để vẽ "người treo cổ"

Các bạn sẽ sử dụng thư viện đã được cung cấp sẵn tại Kho github cho môn CSLT của thầy. Khi vào trang web này, các em kéo xuống dưới để đọc Hướng dẫn cách lấy code, cách thêm thư viện vào project của các em, cách sử dụng thư viện này.

Menu chương trình

Trong các chương trình có nhiều chức năng, thì người lập trình thường hỗ trợ một menu chương trình. Người sử dụng chỉ việc chọn một số hoặc một chữ cái để thực hiện một chức năng nào đó. Khi một chức năng nào đó đã thực hiện xong thì menu đó lại được in ra lại và chương trình hỏi người sử dụng tiếp tục muốn thực hiện chức năng nào. Chương trình chỉ kết thúc/dừng khi người sử dụng chọn số hoặc chữ cái tương đương chức năng dừng chương trình.

Ví dụ menu chương trình có thể như sau khi chương trình chạy :

```
HANGMAN
1 - Choi van moi
2 - Xem danh sach vinh danh
3 - Thoat
Ban chon so may : _
```

Để hỗ trợ tính năng này, người ta thường dùng vòng lặp `while`. Và khi người sử dụng nhập số là 3 (theo ví dụ trên) thì mới ngắt vòng lặp `while`. *Xem slide 23, chủ đề 5 - Các cấu trúc điều khiển (phần 2) để biết cách tạo một menu như trên.*

Cho phép người chơi đoán luôn cả từ

Trong thuật toán trong mục **Cho người chơi chơi 1 ván mới**, tại bước 6, thay vì bạn cho người chơi chỉ nhập vào 1 ký tự thì lúc này bạn cho phép người dùng nhập vào nhiều ký tự. Sau đó, bạn tách các ký tự này ra để đưa vào `letters_guessed`.

Để làm việc này, các bạn đơn giản lặp qua từ được nhập vào, lấy từng ký tự một và thêm vào `letters_guessed`.

In ra những ký tự người chơi chưa đoán được

Phần này được dùng đến khi người chơi đã đoán đủ 6 lần và đoán sai. Lúc này, chương trình cần in ra những ký tự mà người dùng chưa đoán được. Để in các ký tự này, các bạn cho lặp qua các ký tự trong `secret_word`. Nếu ký tự nào không có trong `letters_guessed` thì in ký tự đó ra.

Hỗ trợ đối số dòng lệnh

Xem slide 37-40, chủ đề 6.

Với chương trình này, các bạn có thể hỗ trợ 3 dạng đối số dòng lệnh sau :

```
1 hangman.exe
2 hangman.exe <path>
3 hangman.exe -h
```

Với `hangman.exe` là tên chương trình.

Dạng thứ nhất chỉ gọi chương trình mà không đưa vào đối số nào cả. Như vậy, chương trình sẽ in ra menu chương trình và chờ người sử dụng lựa chọn chức năng tiếp theo để thực hiện.

Dạng thứ hai là gọi chương trình kèm theo đường dẫn (`<path>`) đến tập tin văn bản chứa từ điển.

Dạng thứ tư kèm theo đối số `-h` để yêu cầu in ra hướng dẫn sử dụng của chương trình.

Thuật toán của phần xử lý đối số dòng lệnh như sau :

```
Kiểm tra số lượng đối số dòng lệnh
    Nếu lớn hơn 2 thì thông báo lỗi và in ra hướng dẫn sử dụng chương trình
    Nếu không có đối số nào thì chạy chương trình bình thường
    Nếu có 1 đối số
        Đối số đó là chuỗi -h (so sánh chuỗi ký tự trong đối số với
```

chuỗi ‘‘-h’’) thì in ra hướng dẫn sử dụng
Nếu thuộc dạng 2 thì thay đường dẫn đến file *words.txt* bằng đường dẫn người dùng đưa vào

Tính thời gian chơi 1 ván

Để tính thời gian chơi 1 ván, chúng ta sẽ sử dụng các kiểu dữ liệu `DateTime` và `TimeSpan`. Các bạn xem hướng dẫn lớp `DateTime` tại link 1. Xem hướng dẫn lớp `TimeSpan` tại link 2.

Khi người chơi bắt đầu chơi, bạn dùng thuộc tính `Now` của `DateTime` để lấy ra thời gian lúc mới bắt đầu chơi. Khi người chơi thắng và kết thúc ván chơi, bạn lại dùng thuộc tính `Now` của `DateTime` để lấy thời gian tại lúc đó. Hai đại lượng thời gian này trừ cho nhau sẽ trả ra giá trị kiểu `TimeSpan`. Các bạn lấy số giây thông qua thuộc tính `TotalSeconds` của `TimeSpan`.

Nội dung tập tin văn bản lưu trữ danh sách 10 người chơi thắng trong thời gian ngắn nhất

Nội dung của tập tin này *có thể* theo cấu trúc như sau :

- Mỗi dòng chứa dữ liệu cho một người chơi, bao gồm 2 thành phần phân tách với nhau bằng dấu gạch đứng (|)
  - thành phần thứ nhất là tên người chơi
  - thành phần thứ hai là số giây

Ví dụ sau đây là một phần nội dung của tập tin :

```
1 nguyen van a|1
2 le van b|100
3 tran thi c|200
4 hoang viet d|250
```

Cấu trúc dữ liệu để lưu trữ danh sách 10 người chơi thắng trong thời gian ngắn nhất

Danh sách này có thể rỗng.

Khi chương trình chạy, bạn nên đọc danh sách này lên từ tập tin. Bạn sẽ dùng đến nó khi so sánh kết quả của người chơi.

Để lưu trữ danh sách này, bạn có thể dùng 2 mảng 10 thành phần hoặc hai `List` (một `List` kiểu `string`, một `List` kiểu `long`) hoặc một `List` có các thành phần kiểu cấu trúc. Chúng ta có thể định nghĩa cấu trúc như sau :

```
struct ThanhTich
{
    public string ten; // ho ten nguoi choi
    public long sogiay; // thoi gian choi bang giay
};
```

Khi có một người chơi mới có thành tích có thể thêm vào danh sách này (thành tích nhỏ hơn thành tích của người cuối cùng trong danh sách), thì thành tích của người mới phải được thêm vào tại ví trị sao cho danh sách được sắp xếp tăng dần theo thành tích.

Đọc danh sách 10 người chơi thắng trong thời gian ngắn nhất

Ví dụ, lấy cấu trúc dữ liệu để lưu danh sách là một `List` kiểu cấu trúc `ThanhTich`, thuật toán đọc dữ liệu từ tập tin như sau :

Đầu vào : biến kiểu $List < ThanhTich >$ đã được khởi tạo, gọi là $listThanhTich$

Đầu ra : $listThanhTich$ đã đọc dữ liệu từ file

Đọc dòng đầu tiên

Nếu đọc được dữ liệu (1 dòng văn bản)

Khởi tạo một biến kiểu $ThanhTich$, gọi là $thanhtich$

Dùng hàm Split tách các thành phần trong chuỗi đọc được gán vào biến $listTT$

Nếu $listTT$ không đủ 2 thành phần

Thông báo file dữ liệu bị lỗi, thoát khỏi thuật toán

Ngược lại,

Gán thành phần đầu tiên trong $listTT$ vào thành phần ten của biến $thanhtich$

Gán thành phần thứ hai trong $listTT$ vào thành phần $saogiay$ của biến $thanhtich$

Gán biến $thanhtich$ vào $listThanhTich$

Đọc dòng tiếp theo trong file, quay lại bước 2 của thuật toán này

Lưu danh sách 10 người chơi thắng trong thời gian ngắn nhất

Thuật toán lưu danh sách 10 người chơi thắng trong thời gian ngắn nhất vào file như sau :

Đầu vào : biến kiểu $List < ThanhTich >$ đã được khởi tạo hoặc đã có dữ liệu, gọi là $listThanhTich$

Đầu ra : file văn bản chứa dữ liệu các thành tích

Duyệt qua từng thành tích trong $listThanhTich$, với mỗi thành tích

Sử dụng cú pháp token giữ chỗ để tạo ra một chuỗi ký tự chứa 2 thành phần (tên, số giây) theo cú pháp của file dữ liệu đã quy định, các thành phần phân tách nhau bởi dấu gạch đứng.
Ghi chuỗi ký tự được tạo ra vào file.

Chúc các em làm bài tốt !