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 sáu, ngày 09 tháng 12 năm 2016***
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 một chương trình hoàn chỉnh cho phép quản lý các flashcard và học từ vựng.

Ứng dụng flashcard

Flashcard là phương pháp ghi nhớ kiến thức dựa vào những tấm giấy gợi ý vô cùng hữu ích. Chúng ta có thể tự làm những bộ flashcard cho mình chỉ với giấy, bút và kéo. Nhưng một ứng dụng trong máy tính sẽ giúp cho việc tạo và dùng flashcard được đa dạng, dễ dàng, thuận tiện và hiệu quả hơn.

Một ứng dụng flashcard như vậy sẽ cho phép người dùng nhập các flashcard mới vào. Thao tác nhập này có thể được nhập thủ công hoặc được đọc tự động từ tập tin văn bản. Mỗi flashcard sẽ bao gồm hai thành phần là: *từ* và *nghĩa của từ*.

Khi chuyển sang chế độ **Học**, dựa trên số từ cần học được người dùng nhập vào, ứng dụng sẽ lựa chọn flashcard từ hai nhóm: *flashcard chưa học* và *flashcard đã kiểm tra nhưng chưa thuộc*, theo tỷ lệ 2/3 và 1/3 số từ cần học. Nếu không đủ số lượng flashcard trong mỗi nhóm thì ứng dụng sẽ lấy flashcard của nhóm còn lại hoặc lấy cả từ nhóm *flashcard đã thuộc*. Ứng dụng sẽ hiển thị lần lượt từng nhóm *từ* và *nghĩa của từ* một lên màn hình cho người dùng học.

Khi chuyển sang chế độ **Kiểm tra**, ứng dụng sẽ bóc chọn 10 flashcard từ hai nhóm: *đã học nhưng chưa kiểm tra* và *đã kiểm tra nhưng chưa thuộc*, mỗi nhóm 5 flashcard. Nếu số lượng trong mỗi nhóm không đủ số thì chấp nhận lấy ít hơn 5 flashcard. Ứng dụng hiển thị *nghĩa của từ* từng từ một, người dùng sẽ nhập từ vào. Nếu đúng thì flashcard đó sẽ chuyển thành *flashcard đã thuộc*. Nếu sai thì flashcard đó sẽ chuyển thành *flashcard chưa thuộc*.

Ứng dụng flashcard cũng sẽ có thêm những chức năng sau:

- In ra danh sách tất cả các flashcard
- Thống kê và in các từ đã học
- Thống kê và in các từ đã thuộc

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 ứng dụng flashcard được mô tả như trên. Chương trình cần có các chức năng sau :

1. Có cấu trúc dữ liệu để lưu trữ danh sách các flashcard
2. Thêm một flashcard mới
3. In thông tin một flashcard
4. Liệt kê tất cả các flashcard
5. Tìm kiếm một flashcard dựa vào *từ*
6. Xoá một flashcard
7. Học từ
8. Kiểm tra
9. Thống kê và in các từ đã học
10. Thống kê và in các từ đã thuộc
11. (Tuỳ chọn - cộng điểm) Menu chương trình (Xem ví dụ tại slide 21-22, chủ đề 5 - Các cấu trúc điều khiển 2)
12. (Tuỳ chọn - cộng điểm) Hỗ trợ đối số dòng lệnh (Xem các slide 37-40, chủ đề 7 - Mảng)
13. (Tuỳ chọn - cộng điểm) Lưu danh sách các flashcard vào file
14. (Tuỳ chọn - cộng điểm) Đọc danh sách các flashcard từ file (Xem các slide 14-19, chủ đề 8 - Các cấu trúc dữ liệu khác)

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 nhập dữ liệu vào, thì nên viết đoạn code in dữ liệu ra để kiểm soát việc nhập 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 bộ test sau đây để kiểm tra chương trình và chụp ảnh màn hình kết quả để dán vào báo cáo.

1. Nhập dữ liệu các flashcard sau :

Từ	**Nghĩa**
cultivation	the process of preparing the land for growing crops
irrigation	a means of supplying water for agriculture
precipitation	water that falls to the Earth's surface
a famine	severe hunger; a drastic food shortage
catastrophic	extremely harmful
to abandon	to leave, to give up
to fertilize	to supply nourishment to plants
photosynthesis	the process by which plants combine light energy into chemical energy
to collide	to come together with great or violent force
an eruption	a sudden, often violent, outburst
a flood	an overflowing of water; an excessive amount
to plunge	to go down suddenly
combustion	burning
solar	relating to the sun
an emission	a substance discharged into the air
to convey	to transport from one place to another
a source	a point of origin
a constraint	a restriction
stable	firm and dependable

astrological	related to the study of the position of stars and their supposed effect on earthly events
divination	foretelling the future by finding patterns in physical objects
self-perpetuating	having the power to renew oneself
to haunt	to continually appear as a ghost
to invoke	to call on for support
a phantom	a ghost
a generation	a group of people born at about the same time
inherent	a natural characteristic of something
a process	a series of steps leading to a result
to survive	to continue living
acquisition	the act of taking possession of something

2. Liệt kê tất cả các flashcard
3. Tìm kiếm flashcard có từ *solar*
4. Xoá flashcard có từ *solar*
5. Thêm flashcard mới cho từ *a gap* với nghĩa là *an opening, a distance between two things*
6. Học 20 từ
7. Kiểm tra 10 từ
8. In thống kê các từ đã học
9. In thống kê các từ đã thuộc

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 hơn.

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. Các 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 danh sách các flashcard

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 thông tin *các* flashcard. Mỗi flashcard cần lưu trữ các thông tin: từ, nghĩa của từ, đã học hay chưa, đã kiểm tra hay chưa và đã thuộc hay chưa. Như vậy, cấu trúc dữ liệu để lưu danh sách các flashcard trên phải thoả mãn hai đặc điểm :

1. cho phép lưu trữ một loạt các flashcard
2. mỗi thành phần trong loạt số liệu đó bao gồm :
    - một chuỗi ký tự để lưu từ
    - một chuỗi ký tự để lưu nghĩa của từ
    - một biến kiểu true/false để lưu trạng thái đã học hay chưa
    - một biến kiểu true/false để lưu trạng thái đã kiểm tra hay chưa
    - một biến kiểu true/false để lưu trạng thái đã thuộc hay chưa

Với đặc điểm đầu tiên, bạn có thể dùng mảng hoặc `List` để lưu trữ một loạt các flashcard. 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.

Với đặc điểm thứ hai, chỉ một mảng một chiều hoặc một `List` sẽ không thể nào phù hợp. Do đó, ta có thể sử dụng :

- *Cách 1* : nhiều mảng một chiều hoặc nhiều `List`. Mỗi mảng/`List` có kiểu dữ liệu phù hợp để lưu trữ cho *một* thông tin của flashcard. Trong đó, thành phần thứ i trong mỗi mảng/`List` đó tương ứng với flashcard thứ i.
  Ví dụ (dùng nhiều mảng một chiều): Chúng ta sẽ dùng năm mảng một chiều : hai mảng kiểu `string` để lưu *từ* và *nghĩa của từ*, ba mảng kiểu `bool` để lưu trạng thái đã học hay chưa, đã kiểm tra hay chưa và đã thuộc hay chưa.

- *Cách 2* : một mảng một chiều hoặc một `List` mà mỗi phần tử là một kiểu cấu trúc (`struct`) (Xem slide 5-13, chủ đề 8 - Các cấu trúc dữ liệu khác). Cấu trúc nên được định nghĩa tốt để lưu trữ các thông tin của một flashcard.
  Ví dụ, chúng ta có thể định nghĩa cấu trúc như sau :

```
struct Flashcard
{
    public string tu; // tu
    public string nghia; // nghia cua tu
    public bool dahoc; // da hoc hay chua
    public bool dakiemtra; // da kiem tra hay chua
    public bool dathuoc; // da thuoc hay chua
};
```

Thêm flashcard mới

Trong việc nhập dữ liệu, bạn có thể áp dụng theo hai kiểu sau :

1. *Biết trước* số lượng môn học sẽ được nhập

   Tức là, bạn sẽ yêu cầu người sử dụng nhập vào số lượng môn học muốn nhập. Sau đó, dựa vào giá trị số lượng đó, bạn có thể dùng một vòng lặp để cho phép người sử dụng nhập vào *từ* và *nghĩa của từ* của từng flashcard.

2. *Không biết trước* số lượng môn học sẽ được nhập

   Tức là, số lượng môn học không được biết trước (chương trình không hỏi giá trị này), nên sau khi người sử dụng nhập xong từ và nghĩa của từ của một flashcard thì chương trình sẽ hỏi xem người sử dụng có muốn nhập tiếp hay không. Tuỳ vào câu trả lời của người sử dụng mà chương trình cho phép người sử dụng tiếp tục nhập hoặc thoát ra khỏi tiến trình nhập này. Xem ví dụ tại slide 21-22, chủ đề 5 - Các cấu trúc điều khiển 2.

   Hoặc một cách khác, các bạn có thể dừng việc nhập khi người sử dụng nhập vào ký tự kết thúc file (`end of file - Ctrl + Z`). Xem chủ đề - **Xử lý ký tự ^D cho câu 1, phần Tổng hợp, bài thực hành 3** trong mail group của môn học.

Ví dụ - thuật toán đoạn chương trình nhập dữ liệu theo dạng không biết trước số lượng flashcard sẽ được nhập, cấu trúc dữ liệu theo cách 2 :

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

Đầu ra : $listFlashCard$ đã thêm vào các flashcard mới

Khai báo một biến kiểu $FlashCard$, gọi là $flashcard$

Cho người sử dụng nhập vào ***từ*** mới

Cho người sử dụng nhập vào ***nghĩa của từ*** trên

Gán ***từ*** cho thành phần tu của biến $flashcard$

Gán ***nghĩa của từ*** cho thành phần $nghia$ của biến $flashcard$

Gán $false$ cho thành phần $dahoc$, $dakiemtra$ và $dathuoc$ của biến $flashcard$
Gán biến $flashcard$ vào $listFlashCard$
Hỏi người sử dụng có muốn nhập tiếp flashcard mới hay không
Nếu có, thì trở lại bước đầu tiên
Ngược lại, thoát ra khỏi thuật toán này

Sau đoạn chương trình nhập dữ liệu, bạn nên có đoạn chương trình cho phép in ra dữ liệu đã được nhập vào để kiểm tra lại dữ liệu đã nhập vào có đúng không.

Liệt kê tất cả các flashcard

Bạn dùng một vòng lặp duyệt qua tất cả các flashcard trong $listFlashCard$. Với mỗi flashcard bạn cần in ra các thông tin sau:

- *từ*
- *nghĩa của từ*
- đã học hay chưa, nếu đã học thì đã kiểm tra hay chưa, nếu đã kiểm tra thì đã thuộc hay chưa

Việc in thông tin mỗi flashcard này sẽ được sử dụng lại tại nhiều đoạn chương trình khác. Do đó, bạn nên viết một hàm riêng cho chức năng này với đầu vào là *index* (vị trí flashcard cần in trong trong $listFlashCard$).

Tìm kiếm flashcard dựa vào *từ*

Với cấu trúc dữ liệu theo cách 2, thuật toán tìm kiếm theo *từ* như sau :

Đầu vào : ***từ*** cần tìm
Đầu ra : i - vị trí của flashcard tìm được trong danh sách $listFlashCard$
, bằng -1 nếu không tìm được
Khởi gán $i = 0$
Nếu $i >$ số lượng phần tử trong danh sách $listFlashCard$
Thông báo là không có môn học cần tìm
Trả ra giá trị $i = -1$
Ngược lại,
Nếu phần tử thứ i trong danh sách có biến thành phần tu bằng
với ***từ*** cần tìm
Kết thúc, giá trị của i là vị trí flashcard cần tìm trong danh sách
Ngược lại, tăng giá trị của i lên một đơn vị
Quay lại bước 2 trong thuật toán này

Xoá một flashcard

Tương tự thuật toán sửa thông tin một môn học, thuật toán xoá một môn học như sau :

Yêu cầu người sử dụng nhập vào *từ* muốn xoá
Sử dụng thuật toán *tìm kiếm flashcard theo từ* để lấy vị trí
của flashcard trong danh sách
Nếu kết quả của thuật toán tìm kiếm là -1 ($i == -1$),
    Thông báo không có flashcard cần tìm, yêu cầu nhập lại nếu muốn
    Nếu có nhập lại thì quay lại bước 1 trong thuật toán này
    Nếu không thì thoát khỏi thuật toán
Ngược lại,
    In thông tin của flashcard ra màn hình
    Xác nhận yêu cầu xoá của người sử dụng
    Nếu người sử dụng xác nhận xoá thì thực hiện xoá
    Nếu không, thì thoát khỏi thuật toán

Thuật toán chọn flashcard chưa học

Bốn thuật toán chọn flashcard dưới đây sẽ được hai tính năng Học và Kiểm tra sử dụng. Do đó, bạn nên viết thành các hàm.

Thuật toán này nhận vào số lượng flashcard cần chọn (thuộc loại flashcard chưa học) và trả ra các flashcard đó trong danh sách $listChon$. Đồng thời, trong trường hợp không đủ flashcard thuộc loại chưa học theo yêu cầu, thuật toán sẽ trả ra số lượng còn thiếu đó.

Thuật toán chọn flashcard chưa học như sau:

Đầu vào : $soChuaHoc$ là số flashcard chưa học cần chọn
Đầu ra : Danh sách các flashcard được chọn $listChon$
    và số flashcard còn thiếu $thieu$
Tạo biến $demChuaHoc$ với giá trị khởi tạo là 0 để đếm số flashcard đã được đưa vào danh sách $listChon$
Duyệt qua từng flashcard trong $listFlashCard$
    Nếu $demChuaHoc$ < $soChuaHoc$
        Nếu thành phần $dahoc$ của flashcard có giá trị là $false$ thì gán flashcard đó vào $listChon$
        Tăng biến $demChuaHoc$ lên một đơn vị, quay lại bước 2 để duyệt tiếp qua flashcard khác
Nếu $demChuaHoc$ < $soChuaHoc$
    $thieu = soChuaHoc - demChuaHoc$

Thuật toán chọn flashcard đã học nhưng chưa kiểm tra

Tương tự như thuật toán chọn flashcard chưa học, thuật toán chọn flashcard đã học nhưng chưa kiểm tra như sau:

Đầu vào : $soChuaKT$ là số flashcard đã học nhưng chưa kiểm tra cần chọn
Đầu ra : Danh sách các flashcard được chọn $listChon$
và số flashcard còn thiếu $thieu$
Tạo biến $demChuaKT$ với giá trị khởi tạo là 0 để đếm số flashcard đã được đưa vào danh sách $listChon$
Duyệt qua từng flashcard trong $listFlashCard$
Nếu $demChuaKT < soChuaKT$
Nếu thành phần $dahoc$ có giá trị là $true$ và thành phần $dakiemtra$ có
giá trị là $false$ thì gán flashcard đó vào $listChon$
Tăng biến $demChuaKT$ lên một đơn vị, quay lại bước 2 để duyệt tiếp qua flashcard khác
Nếu $demChuaKT < soChuaKT$
$thieu = soChuaKT - demChuaKT$

Thuật toán chọn flashcard đã kiểm tra nhưng chưa thuộc

Thuật toán chọn flashcard đã kiểm tra nhưng chưa thuộc như sau:

Đầu vào : $soChuaThuoc$ là số flashcard đã kiểm tra nhưng chưa thuộc cần chọn
Đầu ra : Danh sách các flashcard được chọn $listChon$
và số flashcard còn thiếu $thieu$
Tạo biến $demChuaThuoc$ với giá trị khởi tạo là 0 để đếm số flashcard đã được đưa vào danh sách $listChon$
Duyệt qua từng flashcard trong $listFlashCard$
Nếu $demChuaThuoc < soChuaThuoc$
Nếu thành phần $dakiemtra$ có giá trị là $true$ và thành phần $dathuoc$ có
giá trị là $false$ thì gán flashcard đó vào $listChon$
Tăng biến $demChuaThuoc$ lên một đơn vị, quay lại bước 2 để duyệt tiếp qua flashcard khác
Nếu $demChuaThuoc < soChuaThuoc$
$thieu = soChuaThuoc - demChuaThuoc$

Thuật toán chọn flashcard đã thuộc

Thuật toán chọn flashcard đã thuộc như sau:

Đầu vào : $soDaThuoc$ là số flashcard đã thuộc cần chọn
Đầu ra : Danh sách các flashcard được chọn để học $listChon$
và số flashcard còn thiếu $thieu$
Tạo biến $demDaThuoc$ với giá trị khởi tạo là 0 để đếm số flashcard đã được đưa vào danh sách $listChon$
Duyệt qua từng flashcard trong $listFlashCard$
Nếu $demDaThuoc < soDaThuoc$
Nếu thành phần $dathuoc$ có giá trị là $true$ thì gán flashcard đó vào $listHoc$
Tăng biến $demDaThuoc$ lên một đơn vị, quay lại bước 2 để duyệt tiếp qua flashcard khác
Nếu $demDaThuoc < soDaThuoc$
$thieu = soDaThuoc - demDaThuoc$

Học từ

Khi người dùng chọn Học, ứng dụng sẽ cho phép người dùng lựa chọn số lượng từ để học trong khoảng từ 10 đến 20 flashcard. Dựa vào số lượng đó, ứng dụng sẽ bóc chọn 2/3 số flashcard chưa học, 1/3 số flashcard đã học nhưng chưa thuộc. Nếu không đủ số lượng flashcard cho loại chưa học thì số còn thiếu sẽ lấy từ flashcard đã học nhưng chưa thuộc. Tiếp đó, nếu số lượng flashcard đã học nhưng chưa thuộc cũng không đủ thì sẽ lấy cả từ các flashcard đã thuộc.

Thuật toán bóc chọn các flashcard để học cụ thể như sau:

Đầu vào : Danh sách các flashcard $listFlashCard$
Đầu ra : Danh sách các flashcard được chọn để học $listHoc$
Đề nghị người dùng nhập vào số từ cần học trong khoảng từ 10 - 20 flashcard lưu vào biến $sotu$
Nếu người dùng nhập ngoài khoảng trên thì yêu cầu nhập lại
Nếu đúng,
Tạo biến $soChuaHoc$ với giá trị khởi tạo là $2/3 * sotu$
Tạo biến $soChuaThuoc$ với giá trị khởi tạo là $sotu - soChuaHoc$
Dùng ***thuật toán chọn flashcard chưa học*** để lấy các flashcard với đầu vào là $soChuaHoc$, lấy ra danh sách các flashcard trong $listHoc$ và số flashcard thiếu trong $thieu$
Nếu có thiếu flashcard thì cộng dồn số thiếu đó vào biến $soChuaThuoc$
Dùng ***thuật toán chọn flashcard đã kiểm tra nhưng chưa thuộc*** để lấy các flashcard với đầu vào là $soChuaThuoc$, lấy ra danh sách các flashcard trong $listHoc$ và số flashcard thiếu trong $thieu$
Nếu có thiếu flashcard thì dùng ***thuật toán chọn flashcard đã học nhưng chưa kiểm tra*** với đầu vào là $thieu$

Trả *listHoc* ra

Sau khi bóc chọn flashcard xong, ứng dụng sẽ duyệt qua lần lượt từng flashcard trong danh sách được chọn đó, hiển thị thông tin từng flashcard để cho người dùng học.

Kiểm tra

Khi người dùng chọn Kiểm tra, ứng dụng sẽ bóc 5 flashcard từ nhóm *đã kiểm tra nhưng chưa thuộc* và 5 flashcard từ nhóm *đã học nhưng chưa kiểm tra*. Tiếp đến, ứng dụng lần lượt in ra màn hình *nghĩa của từ*, người dùng sẽ nhập *từ*. Nếu người dùng nhập đúng, ứng dụng cộng điểm cho người dùng. Sau khi kết thúc kiểm tra, ứng dụng in ra số điểm mà người dùng đạt được.

Thuật toán chức năng Kiểm tra như sau:

Sử dụng ***thuật toán chọn flashcard đã học nhưng chưa kiểm tra*** để chọn 5 flashcard lưu vào biến *listKT*
Sử dụng ***thuật toán chọn flashcard đã kiểm tra nhưng chưa thuộc*** để chọn 5 flashcard lưu vào biến *listKT*
Khởi tạo biến $demDung = 0$ để đếm số câu trả lời đúng
Duyệt qua từng flashcard trong *listKT*
    In ***nghĩa của từ*** ra màn hình
    Yêu cầu người dùng nhập ***từ*** vào
    Nếu đúng, tăng biến *demDung* lên một
        gán *true* cho thành phần *dakiemtra* và *dathuoc*
    Nếu sai, chỉ gán *true* cho thành phần *dakiemtra*
In biến *demDung* để thông báo số flashcard đã thuộc

Thống kê và in các từ đã học

Duyệt qua *listFlashCard*, tính tổng số flashcard đã học và in các flashcard đã học đó ra màn hình.

Thống kê và in các từ đã thuộc

Duyệt qua *listFlashCard*, tính tổng số flashcard đã thuộc và in các flashcard đã thuộc đó ra màn hình.

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 :

```
UNG DUNG FLASHCARD
1 - Them flashcard moi
2 - Them flashcard moi tu file
3 - Tim kiem
4 - Xoa flashcard
5 - Hoc
6 - Kiem tra
7 - In danh sach flashcard
8 - Thong ke cac flashcard da hoc
9 - Thong ke cac flashcard da thuoc
10 - Doc danh sach flashcard tu file
11 - Ghi danh sach flashcard ra file
12 - 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 12 (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.*

Xử lý đối số dòng lệnh

Xem slide 37-40, chủ đề 6.

Với chương trình này, các bạn có thể hỗ trợ 4 dạng đối số dòng lệnh sau :

```
1 flashcard.exe
2 flashcard.exe <num>
3 flashcard.exe <path>
4 flashcard.exe -h
```

Với `flashcard.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 gọi chương trình kèm với một số (`<num>`). Chương trình sẽ thực hiện chức năng nhập `<num>` flashcard mới. Đây là cách nhập dữ liệu khi biết trước số lượng flashcard được nhập.

Dạng thứ ba là gọi chương trình kèm theo đường dẫn (`<path>`) đến tập tin văn bản chứa các flashcard mới cần nhập. (Đọc thêm phần Thêm flashcard mới từ file văn bản bên dưới)

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 3 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 (thử đổi chuỗi thành số) thì cho nhập dữ liệu
        theo dạng biết trước số môn học sẽ được nhập
        Nếu thuộc dạng 3 thì chuyển đến thêm flashcard mới từ file văn
bản
```

Chú ý *: Nếu bạn không làm yêu cầu 13 và 14 thì có thể bỏ qua việc xử lý dạng thứ 3 của đối số dòng lệnh*

Thêm flashcard mới từ file văn bản

Nội dung của tập tin chứa dữ liệu các flashcard mới *có thể* được quy định như sau :

- Mỗi dòng chứa dữ liệu cho một môn học, bao gồm 6 thành phần phân tách với nhau bằng dấu gạch đứng (|)
  - thành phần thứ nhất là từ
  - thành phần thứ hai là nghĩa của từ

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

```
1 cultivation|the process of preparing the land for
    growing crops
2 irrigation|a means of supplying water for agriculture
3 precipitation|water that falls to the Earth's surface
4 a famine|severe hunger; a drastic food shortage
5 catastrophic|extremely harmful
6 to abandon|to leave, to give up
```

Thuật toán thêm flashcard mới từ file cho cấu trúc dữ liệu cách 2 như sau :

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

Đầu ra : *listFlashcard* đã thêm vào các flashcard mới được đọc từ file

```
    Đọc dòng đầu tiên
```

Nếu đọc được dữ liệu (1 dòng văn bản)
    Khởi tạo một biến kiểu $Flashcard$, tên là $flashcard$
    Dùng hàm Split tách các thành phần trong chuỗi đọc được gán vào biến $listTP$
    Nếu $listTP$ không đủ 2 thành phần
        Thông báo file dữ liệu bị lỗi, thoát khỏi thuật toán
    Ngược lại,
        Gán thành phần đầu tiên trong $listTP$ vào thành phần tu của biến $flashcard$
        Gán thành phần thứ hai trong $listTP$ vào thành phần $nghia$ của biến $flashcard$
        Gán giá trị $false$ cho các thành phần $dahoc$, $dakiemtra$ và $dathuoc$ của biến $flashcard$
    Đọc dòng tiếp theo trong file, quay lại bước 2 của thuật toán này

Đọc dữ liệu từ file văn bản

Với một ứng dụng lớn, thông thường người ta sẽ lưu dữ liệu xuống file văn bản để lần sau ứng dụng sẽ sử dụng lại từ dữ liệu của file đó. Trong ứng dụng này, bạn có thể tổ chức một file văn bản có tên là *data.txt* để lưu dữ liệu các flashcard mà ứng dụng đang có.

Nội dung của tập tin này có thể như sau :

- Mỗi dòng chứa dữ liệu cho một flashcard, bao gồm 5 thành phần phân tách với nhau bằng dấu gạch đứng (|)
  - thành phần thứ nhất là từ
  - thành phần thứ hai là nghĩa của từ
  - thành phần thứ ba là giá trị true/false để chi đã học hay chưa
  - thành phần thứ tư là giá trị true/false để chỉ đã kiểm tra hay chưa
  - thành phần thứ năm là giá trị true/false để chi đã thuộc hay chưa

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

```
1 cultivation|the process of preparing the land for
    growing crops|true|false|false
2 irrigation|a means of supplying water for agriculture|
    true|true|false
3 precipitation|water that falls to the Earth's surface|
    true|true|false
```

```
4 a famine|severe hunger; a drastic food shortage|false|
    false|false
5 catastrophic|extremely harmful|false|false|false
6 to abandon|to leave, to give up|true|false|false
```

Thuật toán đọc dữ liệu từ file cho cấu trúc dữ liệu cách 2 như sau :

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

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

Đọc dòng đầu tiên

Nếu đọc được dữ liệu (1 dòng văn bản)

Khởi tạo một biến kiểu $Flashcard$, gọi là $flashcard$

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

Nếu $listTP$ không đủ 5 thành phần

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

Ngược lại,

Gán thành phần đầu tiên trong $listTP$ vào thành phần tu của biến $flashcard$

Gán thành phần thứ hai trong $listTP$ vào thành phần $nghia$ của biến $flashcard$

Chuyển đổi kiểu cho thành phần thứ ba trong $listTP$ và gán vào thành phần $dahoc$ của biến $flashcard$

Chuyển đổi kiểu cho thành phần thứ tư trong $listTP$ và gán vào thành phần $dakiemtra$ của biến $flashcard$

Chuyển đổi kiểu cho thành phần thứ năm trong $listTP$ và gán vào thành

phần $dathuoc$ của biến $flashcard$

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

Lưu dữ liệu ra file văn bản

Thuật toán lưu dữ liệu ra file văn bản như sau :

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

Đầu ra : file văn bản chứa dữ liệu các flashcard

Duyệt qua từng flashcard trong $listFlashcard$, với mỗi flashcard

Sử dụng cú pháp token giữ chỗ để tạo ra một chuỗi ký tự chứa 5 thành phần (từ, nghĩa, giá trị của thành phần $dahoc$, giá trị của thành phần $dakiemtra$ và giá trị của thành phần $dathuoc$) 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 !