File operations

5 - Thao tác với tệp

- **Nhập liệu từ tệp văn bản**
- Xử lý lỗi với tệp
- Kỹ thuật
  - Giới thiệu các thư viện
    <fstream>, <vector>, <algorithm>
  - Xử lý lỗi đơn giản

Nhập liệu từ tệp (file)

- Hangman hiện thời sử dụng danh sách từ cố định
  - Không cho phép đổi từ vựng (ví dụ: chọn lĩnh vực)
  - Mã nguồn chương trình chứa danh sách từ
    - Phải dịch lại chương trình nếu thay đổi từ
- Giải pháp: *Tách mã nguồn và dữ liệu*
  - Dữ liệu lưu ở tệp
  - Chương trình có mã lệnh đọc tệp, đưa dữ liệu vào bộ nhớ (biến)

Top-down: Sửa main để dùng file

```
const int MAX_BAD_GUESSES = 7;
const char DATA_FILE[] = "data/Ogden_Picturable_200.txt";
...
int main () {
   srand(time(0));
   string word = chooseWord(DATA_FILE);
   if (word.length() < 1) {
      cout << "Error reading vocabulary file " << DATA_FILE;
      return -1;
   }
   string guessedWord = string(word.length(), '-');
   ...
```

- Yêu cầu chooseWord chọn từ file
- Báo lỗi và dừng game nếu file có lỗi

Thư viện fstream

- Thư viện C++ làm việc với file
  - http://www.cplusplus.com/reference/fstream/fstream/
- Làm việc với file
  - Phổ biến trong các phần mềm
  - Phức tạp, tỉ mỉ
  - Có nhiều lỗi “không ngờ”
- Học cách sử dụng **<fstream>**
  - Cách nhanh nhất: làm theo bài hướng dẫn (*tutorials*)
  - Ví dụ: http://www.cplusplus.com/doc/tutorial/files/

Tạo file, ghi vào file với ofstream

- Biến kiểu **ofstream** (out file stream)
  - Đại diện cho một tệp có thể ghi được
  - Phương thức **open**: mở file để ghi
  - Ghi văn bản giống như dùng **cout**

```
// thư viện fstream
#include <fstream>

using namespace std;

int main () {
  ofstream myfile;            // khai báo biến kiểu ofstream
  myfile.open("example.txt");  //Mở file example.txt
  myfile << "Writing this to a file.\n"; //Ghi văn bản vào file
  myfile.close();   //Đóng file lại: giải phóng tài nguyên, ghi vào đĩa
  return 0;
}
```

Tạo file, ghi vào file với ofstream

```
#include <iostream>
#include <fstream>
using namespace std;

int main () {
  ofstream myfile ("example.txt");
  if (myfile.is_open()) {     // Kiểm tra việc mở tệp có thành công?
    myfile << "This is a line.\n";
    myfile << "This is another line.\n";
    myfile.close();
  }
  else cout << "Unable to open file";
  return 0;
}
```

Đọc file với ifstream

```
...
#include <fstream>                                                    //Thư viện fstream chứa ifstream
using namespace std;

int main () {
  string line;
  ifstream myfile ("example.txt");   //Mở file example.txt đã ghi ở ví dụ trước
  if (myfile.is_open()) {                                            //Kiểm tra việc mở tệp có thành công ?
    while ( getline (myfile,line) ) { //Hàm getline đọc 1 dòng của tệp vào biến line
      cout << line << '\n';                          //...và chuyển vị trí đọc xuống dòng tiếp theo
    }                                                //  Lặp đến khi getline trả về “false” (tức là không còn gì để đọc, hết tệp)
    myfile.close();                    //Đóng tệp, giải phóng tài nguyên hệ thống
  }
  else cout << "Unable to open file";

  return 0;
}
```

Đọc từ vựng Hangman từ tệp

Từ vựng của Hangman được lưu trong một tệp văn bản:

- Tệp nằm trong thư mục “data” cùng với chương trình (quyết định tại nơi gọi chooseWord, hiện là main())
- Mỗi từ trên một dòng

chooseWord (thử đọc từ file)

```
string chooseWord(const char* fileName)
{
   ifstream file(fileName);   //Mở tệp có đường dẫn như trong tham số
   if (file.is_open()) {      // Kiểm tra tệp mở thành công
      string word;
      while (file >> word) {  //Đọc từng từ đến khi không đọc được nữa
         cout << word << endl;//ghi tạm ra màn hình để xem thử
      }
      file.close();
   } else cout << "Error opening " << fileName;
   return "book";             //  return tạm gì đó để chạy được với main.
}
```

Lưu trữ dữ liệu từ file vào đâu?

Từ vựng của Hangman được lưu trong một tệp văn bản:

- Mỗi từ trên một dòng
  - Số dòng (số từ) chưa biết trước

→ Cần kiểu dữ liệu lưu trữ số lượng từ “tùy ý”

nếu dùng mảng thông thường ta sẽ phải đọc một lần để đếm số dòng trước khi khai báo mảng, sau đó mới đọc vào mảng.

Thư viện vector

- Cho phép lưu trữ dãy giá trị cùng kiểu
  - Truy xuất giống như mảng tĩnh
  - Ví dụ: **x[i]**
- Cho phép thay đổi kích thước (số phần tử)
  - Có thể coi như mảng "động"
  - Không cần tự lập trình xin cấp phát bộ nhớ
- Nhiều tiện ích thao tác với mảng
  - Thêm, chèn, xóa, sửa
  - Kết hợp với **<algorithm>**: tìm kiếm, sắp xếp …

http://www.cplusplus.com/reference/vector/vector/

Thư viện vector

Chèn vào cuối vector

```
// push_back
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;
  int myint;

  cout << "Please enter some integers (Ctrl-D to
end):\n";

  while (cin >> myint) {
    myvector.push_back (myint);
  }

  cout << "myvector stores " <<
int(myvector.size()) << " numbers.\n";

  return 0;
}
```

```
// push_back

Sử dụng thư viện vector

Khai báo myvector là vector các số
nguyên

Lặp đến khi không còn dữ liệu mới
Phương thức push_back: Thêm myint
vào cuối myvector

In số phần tử của myvector
```

Thư viện vector

Truy xuất các phần tử trong vector

```
vector<int> myvector (10);   // 10 zero-initialized ints

// assign some values:
unsigned sz = myvector.size();
for (unsigned i=0; i<sz; i++)
  myvector.at(i)=i;

cout << "myvector contains:";
for (unsigned i=0; i<sz; i++)
  cout << ' ' << myvector.at(i);
cout << '\n';

// reverse vector using operator[]:
for (unsigned i=0; i<sz/2; i++)
{
  int temp;
  temp = myvector[sz-1-i];
  myvector[sz-1-i]=myvector[i];
  myvector[i]=temp;
}

cout << "myvector contains:";
for (unsigned i=0; i<sz; i++)
  cout << ' ' << myvector[i];
cout << '\n';
```

Khai báo vector có 10 phần tử

Lưu kích thước vector

Gán giá trị tại vị trí thứ i (tính từ 0) qua phương thức at

In giá trị tại vị trí thứ i qua phương thức at

Sử dụng toán tử [] truy xuất và gán giá trị phần tử của vector
giống như mảng tĩnh

In giá trị tại vị trí thứ i qua phương thức at

chooseWord (đọc vào vector)

```
string chooseWord(const char* fileName)
{
   vector<string> wordList;        //Khai báo vector chứa các từ sẽ đọc
   ifstream file(fileName);        //Mở tệp có đường dẫn như trong tham số
   if (file.is_open()) {           // Kiểm tra tệp mở thành công
      string word;
      while (file >> word) {       //Đọc từng từ (giống cin) đến khi không đọc được nữa
         wordList.push_back(word);          //đưa từ vừa đọc vào vector
      }
      file.close();
   }
   if (wordList.size() > 0) {                 // nếu có dữ liệu đọc thành công
      int randomIndex = rand() % wordList.size();
      return wordList[randomIndex];           // trả về một từ ngẫu nhiên trong vector
   } else return "";                          // nếu không đọc được gì, trả về từ rỗng
}
```

Cẩn thận trường hợp file mở thành công nhưng rỗng

Hoàn thành Hangman 2.0

- Đọc dữ liệu từ tệp
  - Sử dụng **<fstream>**, **<vector>**
- Lựa chọn phần tử ngẫu nhiên trong **vector**

Chuẩn hóa dữ liệu

Dữ liệu từ tệp, đặc biệt là dữ liệu tải về từ Internet cần được ***chuẩn hóa***

- Đảm bảo chương trình hoạt động với dữ liệu đúng như ý định ban đầu
- Sửa lỗi dữ liệu, loại bỏ dữ liệu "xấu"

Với Hangman 2.1, cần chuyển mọi từ về *dạng chữ thường* để phép toán so sánh (**==, !=**) hoạt động chính xác

chooseWord (chuẩn hóa dữ liệu)

```
string chooseWord(const char* fileName)
{
   vector<string> wordList;
   ifstream file(fileName);
   if (file.is_open()) {
      string word;
      while (file >> word) {
         wordList.push_back(word);
      }
      file.close();
   }
   if (wordList.size() > 0) {
      int randomIndex = rand() % wordList.size();
      return getLowerCaseString(wordList[randomIndex]);
   } else return "";
}
```

Chuyển từ được chọn sang chữ thường trước khi trả về

Chuyển từ sang chữ thường

```
string chooseWord(const char* fileName)
{
    vector<string> wordList;
    ifstream file(fileName);
    if (file.is_open()) {
        string word;
        while (file >> word) {
            wordList.push_back(w
        }
        file.close();
    }
    if (wordList.size() > 0) {
        int randomIndex = rand() % wordList.size();
        return getLowerCaseString(wordList[randomIndex]);
    } else return "";
}
```

```
string getLowerCaseString(const string& s)
{
    string res = s;
    int sz = s.size();
    for (int i = 0; i < sz; i++)
        res[i] = tolower(s[i]);
    return res;
}
```

Giới thiệu thư viện algorithm

```
string getLowerCaseString(
     const string& s)
{
    string res = s;
    int sz = s.size();
    for (int i = 0; i < sz; i++)
        res[i] = tolower(s[i]);
    return res;
}
```

```
#include <algorithm>

string getLowerCaseString(const string& s)
{
    string res = s;
    transform(s.begin(), s.end(), res.begin(), ::tolower);
    return res;
}
```

Duyệt từ đầu đến cuối của **s**, biến đổi bằng hàm **tolower()**, đặt kết quả lần lượt vào các ký tự tính từ đầu của **res**

Duyệt mảng là một thao tác phổ biến nhất trong lập trình

http://stackoverflow.com/questions/313970/how-to-convert-stdstring-to-lower-case

Con trỏ duyệt (Iterator)

s.begin(), s.end()

trả về các ***iterator***

là khái niệm *khái quát hóa* của chỉ số mảng

```
#include <algorithm>

string getLowerCaseString(const string& s)
{
    string res = s;
    transform(s.begin(), s.end(),
res.begin(), ::tolower);
    return res;
}
```

- Sẽ học kỹ hơn ở các buổi sau
- http://www.cplusplus.com/reference/iterator/

Hoàn thành Hangman 2.1

- Chuẩn hóa từ về dạng chữ thường
  - Duyệt mảng, biến đổi sử dụng **<algorithm>**

Bài tập: Hangman 2.2 - Chọn tệp dữ liệu

- Từ tham số dòng lệnh
- Từ lựa chọn của người chơi

Nội dung

- Nhập liệu từ tệp văn bản
- **Xử lý lỗi với tệp**
- Kỹ thuật
  - Thư viện

    <fstream>, <vector>, <algorithm>

Các phiên bản sau

Bạn có thể tự làm tiếp

- 2.2. Cho chơi nhiều lần
- 2.3. Hoạt hình: giá treo cổ lắc lư sau khi thua, nếu thắng thì có một người đứng nhảy múa

Đồ họa? Đợi khi học thư viện đồ họa