**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**



**KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

****

**BÁO CÁO MÔN HỌC**

**Kĩ Thuật Lập Trình**

# Chủ đề : Xây dựng ứng dụng Từ điển BKEncyclopedia

|  |  |  |
| --- | --- | --- |
| Giảng viên hướng dẫn | : | Trịnh Thành Trung |
| Sinh viên thực hiện | : | Nguyễn Anh Tuấn – 20134284  Hà Duy Anh – 20130065  Nguyễn Thị Trà My – 20132632  Lương Công Tiến – 20133939  Đỗ Duy Linh - 20132258 |

**Báo cáo BTL**

**Môn Kĩ thuật lập trình**

**Chương I: Tổng quan**

1. Về hệ thống bách khoa toàn thư BKEncyclopedia

Là một hệ thống cho phép người dùng thêm sửa xóa và tìm kiếm các từ thuộc nhiều thể loại khác nhau.

1. Định hướng công nghệ, công cụ sử dụng để xây dựng.

Hệ thống được viết bằng ngôn ngữ C, giao diện console không sử dụng các thư viện bên ngoài.

**Chương II: Phân tích thiết kế**

1. Mô tả
2. Yêu cầu:

* Gõ vào một từ và hiển thị tất cả các thông tin của từ đó
* Thêm từ
* Thêm thể loại
* Thống kê tất cả các từ trong một thể loại nào đó
* Sửa, xóa từ
* Sửa, xóa thể loại. Tất cả các từ của thể loại đó sẽ bị xóa.
* Thoát chương trình.

1. Input của hệ thống

File thể loại có định dạng như sau

Mã thể loại  
Tên thể loại  
Các trường, phân cách bằng dấu

Mỗi thể loại sẽ có 1 file lưu các từ định dạng như sau

Từ  
Mã thể loại  
Giá trị các trường

1. Các chức năng của hệ thống

* Tìm từ
* Thêm từ
* Sửa từ
* Xóa từ
* Thêm thể loại
* Sửa thể loại
* Xóa thể loại
* Thống kê các từ 1 thể loại
* Thống kê tất cả các từ
* Thoát chương trình

1. Cấu trúc dữ liệu

Thể loại được lưu vào danh sách liên kết đơn có nút đầu rỗng là struct category\_node, có cấu trúc

/\* Nut dau rong \*/

typedef struct category\_node {

char\* category\_id; // Mã thể loại

char\* category\_name; // Tên thể loại

long amount\_of\_type; // So cac truong

long amount\_of\_word; // So cac tu

type\_node\* type\_head; // Con trỏ trỏ tới danh sách liên kết các trường

struct category\_node\* next; // Con trỏ next

} category\_node;

Các trường của mỗi thể loại là 1 danh sách liên kết và node đầu được quản lý bởi 1 node thuộc category\_node

typedef struct type\_node {

char\* type\_name; // Tên thể loại

struct type\_node\* next; // Con trỏ next

} type\_node;

Lý do nhóm em xây dựng cơ sở dữ liệu cho thể loại như thế này là do thể loại không có nhiều nên việc duyệt qua các thể loại không mất nhiều thời gian nếu dùng danh sách liên kết mà lại có lợi thế không dư thừa bộ nhớ.

Các từ được lưu vào mảng động (dynamic array), nhóm em xây dựng cấu trúc dữ liệu mảng động tương tự như vector ở C++. Mảng này có ưu điểm so với mảng thông thường là số lượng phần tử tối đa có thể thay đổi

Cấu trúc

typedef struct Vector {

long used; // Số phần tử của mảng

long capacity; // Số phần tử max

word\_info\*\* word\_list; // Mảng các con trỏ cấu trúc word\_list

} Vector;

Cấu trúc word\_info là các phần tử của mảng word\_list

typedef struct {

data\_node\* p\_data\_head; // Trỏ tới node đầu của danh sách liên kết tới trường của từ

char\* category\_id; // Mã thể loại của từ

char\* word\_name; // Tên từ

} word\_info;

Cấu trúc data\_node là danh sách liên kết lưu nội dung các trường của từ

typedef struct data\_node {

char\* data; // Nội dung 1 trường

struct data\_node\* next; // Con trỏ next

} data\_node;

Nhóm em xây dựng cấu trúc dữ liệu mảng động kết hợp với danh sách liên kết để lưu các từ bởi vì:

* Sử dụng mảng tuy tốn bộ nhớ hơn nhưng có tốc độ tìm kiếm tốt hơn nhiều nếu mảng đã sắp xếp vì không phải duyệt lần lượt. Hơn nữa do số lượng phần tử tối đa có thể tăng lên nên số từ lưu vào không giới hạn.
* Sử dụng danh sách liên kết lưu các trường của từ vì 1 từ không có quá nhiều trường nên việc duyệt phần tử không tốn nhiều thời gian mà tối ưu bộ nhớ.

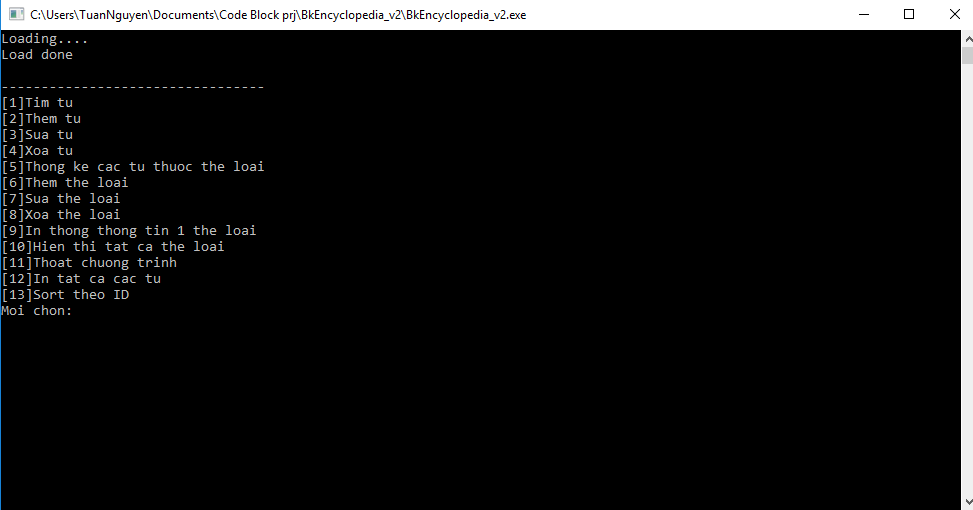
**Chương III: Cài đặt và kiểm thử**

1. Một số code của cấu trúc dữ liệu chúng em xây dựng

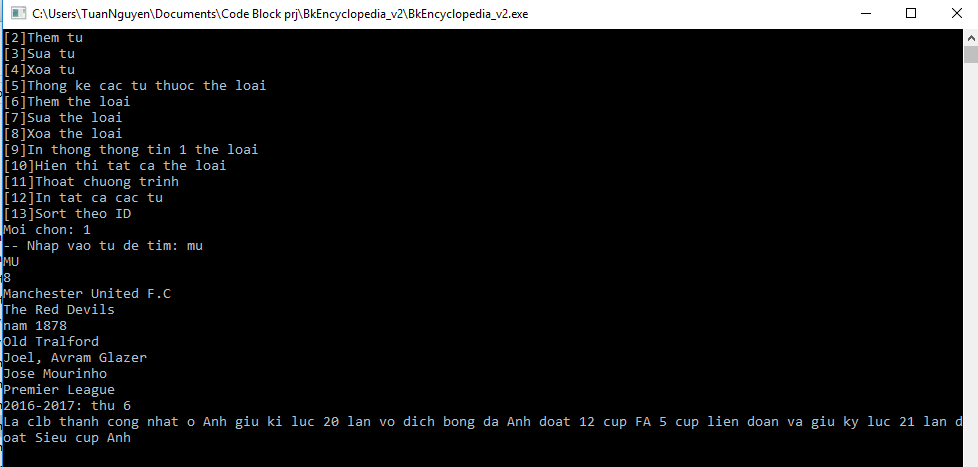
Xem phụ lục 1

1. Các chức năng của chương trình

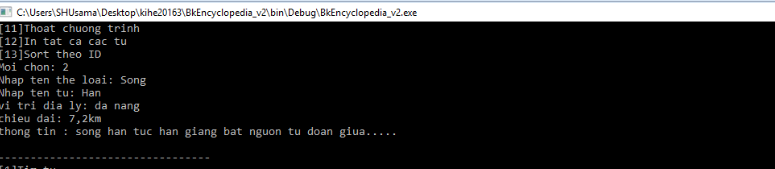
Giao diện chính



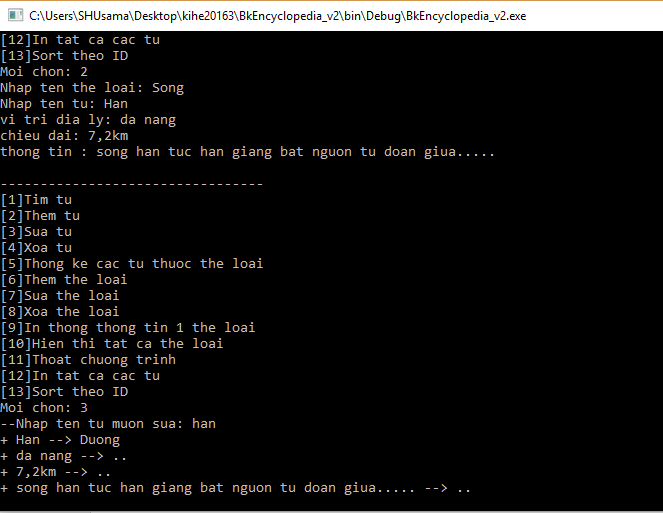
Tìm từ



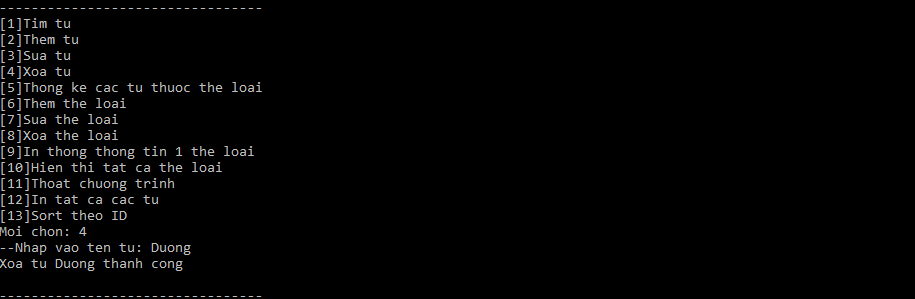
Thêm từ



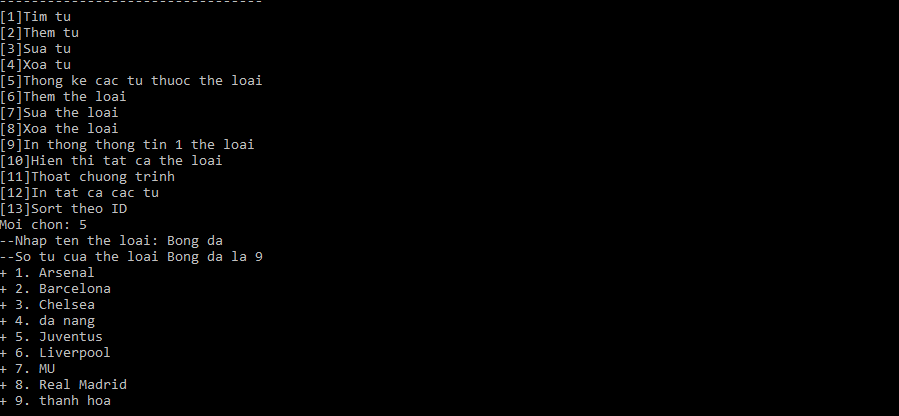
Sửa từ



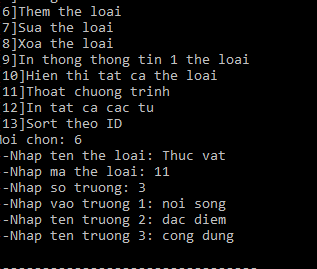
Xóa từ



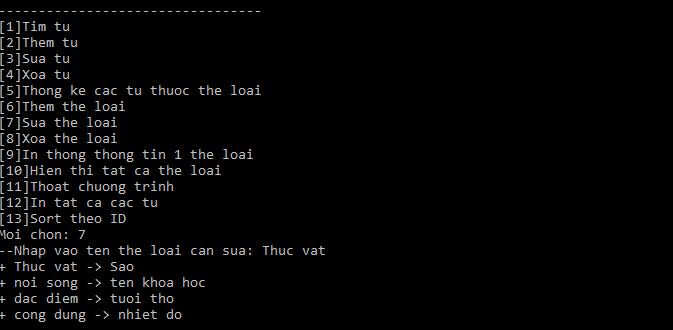
Thống kê các từ của thẻ loại



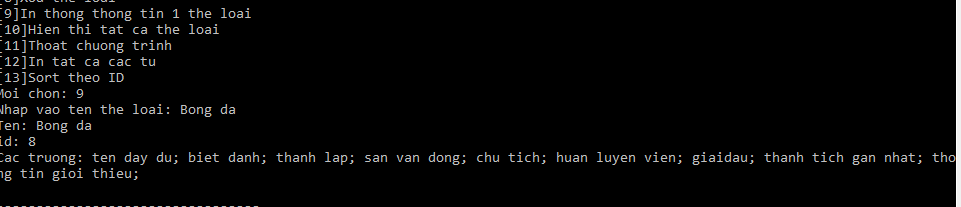
Thêm thể loại



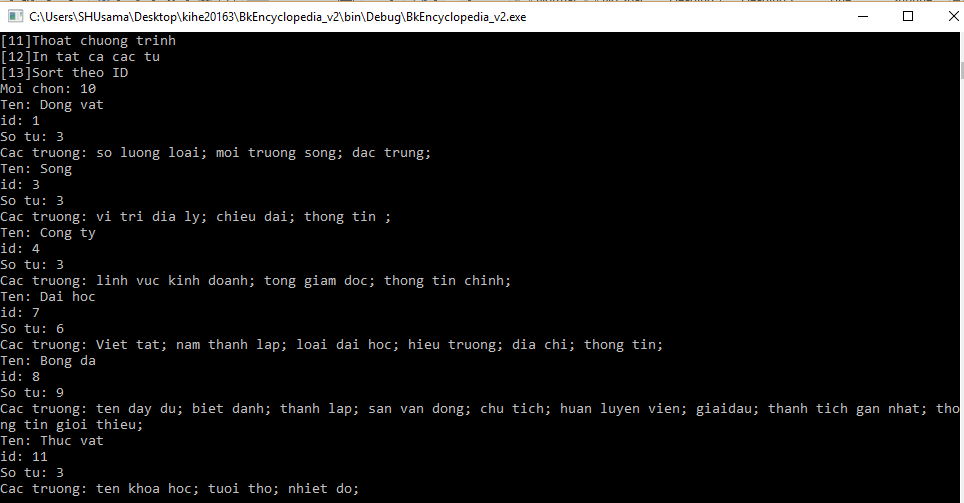
Sửa thể loại



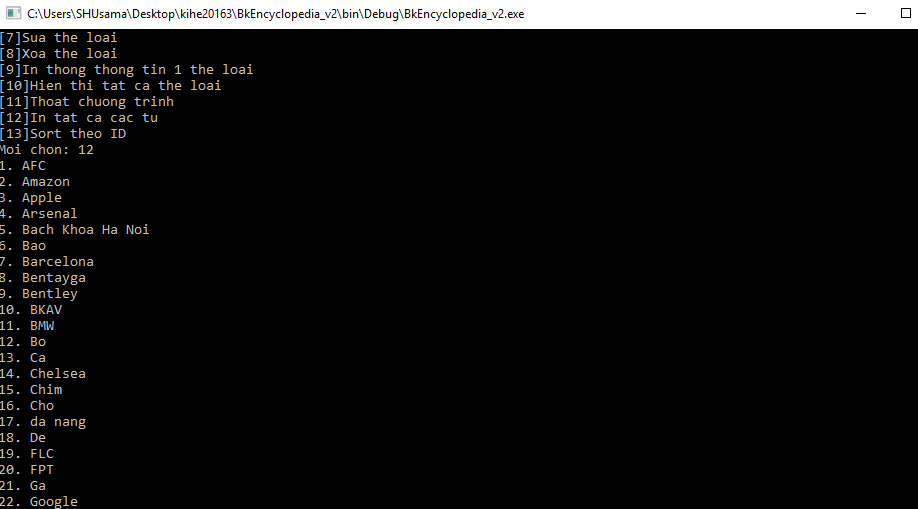
In thông tin 1 thể loại



Hiển thị tất cả thể loại



In tất cả các từ



1. Kiểm thử

Các test case khiến chương trình chạy lỗi

* Format file sai. Do chương trình không chuẩn hóa được file đầu vào nên nếu format file sai thì chương trình đọc sai
* Người dùng nhập quá dài. Ngoại trừ menu chính là có kiểm soát độ dài input của người dùng, khi vào các menu con nếu người dùng nhập quá dài chương trình có lỗi tràn bộ đệm -> crash.

Đánh giá: Chương trình làm việc tốt với dữ liệu lớn (đã test với 1000000 từ) với tốc độ nhanh nhưng còn chưa chuẩn hóa được file dữ liệu đầu vào và kiểm tra kích thước input của người dùng.

**Chương IV: Phụ lục**

1. Một số code về cấu trúc dữ liệu chúng em xây dựng

/\* Khởi tạo vector \*/

int init\_vector(Vector\*\* vector, int size){

\*vector = calloc(1, sizeof(vector));

if (vector == NULL)

return -1;

(\*vector)->capacity = size;

(\*vector)->used = 0;

(\*vector)->word\_list = calloc(size, sizeof(word\_info\*));

if ((\*vector)->word\_list == NULL)

return -1;

return 1;

}

/\* Xoa phan tu lien tiep trong vector \*/

void remove\_many\_element(Vector\* vector, long first\_index, long last\_index){

word\_info\*\* word\_list;

long amount\_of\_word;

long distance;

word\_list = vector->word\_list;

amount\_of\_word = vector->used;

distance = last\_index - first\_index;

for(long i = last\_index + 1; i < amount\_of\_word; i++){

word\_list[i-distance] = word\_list[i];

}

vector->used -= (distance + 1);

}

/\* insert 1 tu vao mang word\_list \*/

int insert\_vector\_element(Vector\* vector, word\_info\* new\_word, long index){

long amount\_of\_word;

if (index < 0 || index > vector->used)

return -1;

if (vector->used == vector->capacity)

resize\_vector(vector);

word\_info\*\* word\_list = vector->word\_list;

amount\_of\_word = vector->used;

/\* Dich trai cac phan tu ke tu index \*/

for (int i = amount\_of\_word-1; i >= index; i--){

vector->word\_list[i+1] = vector->word\_list[i];

}

word\_list[index] = new\_word;

(vector->used)++;

return 1;

}

/\* Free vector \*/

void free\_vector(Vector\* vector){

if (vector == NULL)

return;

free\_word\_list(vector->word\_list, vector->used);

free(vector->word\_list);

free(vector);

}

void free\_word\_list(word\_info\*\* word\_list, long word\_size){

for (long i = 0; i < word\_size; i++){

free(word\_list[i]);

}

}

1. Phân công công việc trong nhóm

* Nguyễn Anh Tuấn (nhóm trưởng): Xây dựng cơ sở dữ liệu cho chương trình, cung cấp các hàm để làm việc với cơ sở dữ liệu đã xây dựng. Thực hiện việc đọc dữ liệu từ file và đưa vào cơ sở dữ liệu đã xây dựng. Khi kết thúc chương trình giải phóng vùng nhớ đã cấp phát
* Hà Duy Anh: Xây dựng hàm thêm, sửa dữ liệu vào file đồng thời phụ trách 3 chức năng lớn thêm từ, sửa từ, thêm thể loại.
* Nguyễn Trà My: Xây dựng thuật toán cho chương trình, sắp xếp các từ theo tên và theo mã thể loại (dùng quick sort) và thuật toán tìm từ theo tên (binary search).
* Lương Công Tiến: Xây dựng các hàm lấy ra tất cả thông tin các từ và thể loại từ cơ sở dữ liệu của chương trình. Thực hiện việc in thông tin từ, thể loại và chức năng sửa, xóa thể loại.
* Đỗ Duy Linh: Phụ trách chính việc sinh lượng lớn các từ phục vụ cơ sở dữ liệu của chương trình và phần kiểm tra đầu vào. Xây dựng một số hàm để làm việc với chuỗi, kiểm tra file và kiểm thử sản phẩm.