**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



**BÁO CÁO BÀI TẬP LỚN**

**Bộ môn:** C Advance

**Giảng viên:** Nguyễn Bá Ng

**Nhóm thực hiện:**  BIGBOSS

Sinh viên: *Nguyễn Xuân Hải – MSSV: 20121651*

*Nguyễn Huy Hưng – MSSV: 20121866*

*Phạm Tất Đạt MSSV: 20130866*

*Trần Bảo Huy – MSSV: 20133411*

**Mục lục**

1. Cơ sở lý thuyết

## Cấu trúc dữ liệu Btree và thư viện B-Tree

## Thuật toán Soundex

1. Xây dựng chương trình
2. Các chức năng cơ bản
3. Chức năng tra từ
4. Chức năng thêm từ
5. Chức năng xóa từ
6. Chức năng sửa từ
7. Các thư viện sử dụng
8. Thư viện B-tree
9. Thư viện Soundex
10. Thư viện Readline
11. Đánh giá

# Phân công thực hiện

|  |  |
| --- | --- |
| **Thành viên** | **Công việc thực hiện** |
| Nguyễn Xuân Hải | Thực hiện chức năng tìm thêm từ |
| Phạm Tất Đạt | Thực hiện chức năng tìm kiếm |
| Nguyễn Huy Hưng | Thực hiện chức năng cập nhật  Viết báo cáo |
| Trần Bảo Huy | Hoàn thành chức năng xóa từ  Test hiệu năng chương trình |
|  |  |
|  |  |

# I. Cơ sở lý thuyết

## 1. Cấu trúc dữ liệu Btree và thư viện B-Tree

**\* Cấu trúc dữ liệu B-Tree**:

B-Tree là một cây cân bằng nhiều nhánh, tức là có nhiều hơn một khóa cho mỗi nút và tất cả các nút lá được cùng một khoảng cách từ nút gốc. B-Tree duy trì tính cân bằng bằng cách đảm bảo các nút lá đều có cùng độ sâu. [1]

Có thể lưu trữ được số lượng phần tử rất lớn phù hợp với việc lưu trữ trên đĩa.

Cấu trúc dữ liệu mà duy trì một bộ sắp xếp dữ liệu và cho phép hoạt động hiệu quả để tìm kiếm, xóa, chèn và duyệt dữ liệu.

## 2. Thuật toán Soundex [2]

Mã Soundex trả về một xâu bao gồm có một chữ cái và 3 trong số theo sau: chữ cái là chữ cái đầu tiên của từ đó, và các số mã hóa cho các phụ âm còn lại. Các phụ âm có âm sắc giống nhau được mã bởi các trọng số giống nhau, ví dụ, âm lưỡi B, F, P, và V được mã hóa là 1. Các nguyên âm có thể ảnh hưởng đến việc mã hóa, nhưng không bao giờ được mã hóa trực tiếp trừ khi chúng xuất hiện ngay ở đầu của từ.

Thuật toán chi tiết như sau:

b1. Lưu chữ cái đầu tiên trong xâu.

b2. Loại bỏ mọi chữ cái sau đây, trừ khi nó là chữ cái đầu tiên: a, e, h, i, o, u, w, y

b3. Gán các trọng số cho các chữ cái còn lại (sau chữ cái đầu tiên) như sau:

* + b, f, p, v = 1
  + c, g, j, k, q, s, x, z = 2
  + d, t = 3
  + l = 4
  + m, n = 5
  + r = 6

b4. Nếu 2 hay nhiều chữ cái có trọng số giống nhau ở sát nhau trong từ ban đầu (trước bước 1), hay cách nhau chỉ bởi chữ h hay chữ w (chỉ có trong tiếng Mỹ), thì bỏ qua tất cả.

b5. Trả lại 4 ký tự đầu tiên, thêm 0 vào bên phải nếu ít hơn 4 ký tự được trả về.

Sử dụng thuật toán này, cả "Robert" và "Rupert" đều trả về chuỗi "R163" trong khi từ "Rubin" sẽ trả về "R150".

# 

# 

# II. Xây dựng chương trình

## 1. Các chức năng cơ bản

### a) Chức năng tra từ

Trong chương trình, chúng em tạo 2 file B Tree: word.dic (lưu trữ từ + nghĩa của từ) và soundex.dic (lưa trữ từ + mã soundex của từ).

Các bước thưc hiện:

- Định vị vị trí của từ cần tra trong file word.dic

btsel(dic,(char\*) word, mean, 20000, &rsize);

* Nếu hàm trên trả về 0, tức là từ đang tra có trong từ điển, chỉ cần in ra nghĩa của từ:
* Nếu từ đang tra không có trong dữ liệu từ điển, sẽ tìm kiếm gợi ý sử dụng SounDex.

### b) Chức năng thêm từ

Các bước thực hiện:

- Kiểm tra xem đã có từ trong từ điển chưa , nếu có thì hiển thị thông báo là đã tồn tại trong từ điển sau đó thông báo có muốn update không ?.

- Nếu từ chưa có trong từ điển, tiến hành thêm từ và nghĩa của từ vào file word.dic, đồng thời thêm tù và mã soundex vào file soundex.dic.

- Nếu từ đã có trong từ điển, sẽ gợi ý người sử dụng cập nhật cho từ đó.

### c) Xóa từ

Để xóa 1 từ trong từ điển, trước hết cần kiểm tra từ đó có tồn tại trong từ điển hay không, nếu tồn tại, có 2 việc cần làm:

- Xóa từ và nghĩa của từ trong file word.dic

- Xóa từ và mã soundex của từ trong soundex.dic

Nếu không tồn tại thì hiển thì thông báo không tồn tại và xóa

**d) Cập nhật từ**

Thực hiện công việc cập nhật nghĩa cho 1 từ trong từ điển, trước hết cần kiểm tra từ đó có tồn tại trong từ điển hay không:

* Nếu tồn tại : sẽ cập nhật nghĩa của từ trong word.dat
* Nếu không tồn tại sẽ gợi ý người dùng thêm từ đó vào trong từ điển

## 

## 2. Các thư viện sử dụng

**a) Thư viện B Tree**

Các API của thư viện btree.h [4]

|  |  |  |
| --- | --- | --- |
| **STT** | **Lệnh** | **Ý nghĩa** |
| 1 | gcc -o simple simple.c -Iinc -Llib -lbt  hoặc:  gcc -o simple simple.c libbt.a | Lệnh biên dịch |
| 2 | int **btinit**(void); | Khởi tạo thư viện B Tree |
| 3 | BTA\* **btcrt**(char\* fid, int nkeys, int shared); | Tạo file B Tree |
| 4 | BTA\* **btopn**(char\* fid, int mode, int shared); | Mở file B Tree |
| 5 | int **btcls**(BTA\* btact); | Đóng file B Tree |
| 6 | int **btdups**(BTA\* btact, int dups); | Hỗ trợ set/unset cho các khóa trùng lặp |
| 8 | int **binsky**(BTA\* btact, char\* key, BTint value); | Chèn một key |
| 9 | int **bfndky**(BTA\* btact, char\* key, BTint\* value); | Tìm kiếm theo key |
| 10 | int **bnxtky**(BTA\* btact, char\* key, BTint\* value); | Tìm kiếm chuỗi khóa liên tiếp |
| 11 | int **bprvky**(BTA\* btact, char\* key, BTint\* value); | Thứ tự đảo ngược của các khóa |
| 12 | int **btpos**(BTA\* btact, int pos); | thiết lập vị trí bên trong một chỉ mục B Tree |
| 13 | int **bdelky**(BTA\* btact, char\* key); | Xóa một key |
| 14 | int **bupdky**(BTA\* btact, char\* key, BTint value); | Update giá trị của key |
| 15 | int **btcrtr**(BTA\* btact, char\* root); | Tạo root |
| 16 | int **btchgr**(BTA\* btact, char\* root); | Thay đổi root hiện tại |
| 17 | int **btdelr**(BTA\* btact, char\* root); | Xóa root |
| 18 | int **btlock**(BTA\* btact); | Giành quyền truy cập độc quyền cho một tập tin BTree |
| 19 | int **btunlock**(BTA\* btact); | Phát hành truy cập độc quyền vào một tập tin B Tree |
| 20 | int **btins**(BTA\* btact, char\* key, char\* data, int dsize); | Chèn key và dữ liệu |
| 21 | int **btupd**(BTA\* btact, char\* key, char\* data, int dsize); | Cập nhật dữ liệu cho khóa hiện tại |
| 22 | int **btseln**(BTA\* btact, char\* key, char\* data, int dsize, int\* rsize); | Định vị dữ liệu cho khóa tiếp theo trong chuỗi. |
| 23 | int **btselp**(BTA\* btact, char\* key, char\* data, int dsize, int\* rsize); | Định vị dữ liệu cho các khoá trước đó trong chuỗi |
| 24 | int **btrecs**(BTA\* btact, char\* key, int\* rsize); | Xác định kích thước của bản ghi dữ liệu cho khóa cụ thể |
| 25 | int **btdel**(BTA\* btact, char\* key); | Xóa một key và dữ liệu liên quan |
| 26 | int **btsel**(BTA\* btact, char\* key, char\* data, int dsize, int\* rsize); | Định vị dữ liệu cho khóa hiện tại |
| 27 | int **bdbug**(BTA\* btact, char\* opt, BTint blk); | Gỡ rối thông tin |
| 28 | void **btcerr**(int\* ierr, int\* ioerr, char\* srname, char\* msg); | Đọc tin nhắn lỗi |

**b) Thư viện soundex**

Chương trình từ điển này nhóm chúng em sử dụng thư viện soundex để tìm các từ tương đồng (cùng mã soundex).

Trong mã nguồn, thuật toán soundex được lưu trong file soundex.h, với hàm:

int Cal\_SoundEx(char \*SoundEx, char \*WordString, int LengthOption, int CensusOption)

tham số đầu tiên sẽ dùng để lưu trữ mã soundex cho từ WordString, 2 tham số sau thường được thiết lập mặc định là 4 và 1.

**c) Thư viện GNU Readline [5]**

Đây là thư viện mã nguồn mở, thư viện hỗ trợ vđọc sự kiện từ bàn phím để thực hiện các chức năng tự động hoàn thành từ đang viết.

Để sử dụng thư viện, khi biên dịch chỉ cần thêm tùy chọn: -lreadline.

# III. Đánh giá

Trong chương trình từ điển này, chúng em đã tập trung xây dựng theo các yêu cầu của đề bài với mục đích chính là sử dụng thư viện Btree, thư viện soundex để tạo dựng được chương trình từ điển thỏa mãn yêu cầu.

**Các yêu cầu đã làm được:**

- Thao tác tra từ, thêm từ , cập nhật và xóa từ.

- Sử dụng thư viện soundex cho tìm kiếm với gợi ý

- Test hiệu năng chương trình với bộ dữ liệu: 1 – 10 triệu từ

**Các yêu cầu chưa làm được:**

Chưa sử dụng giao diện đồ họa

**Hướng hoàn thiện chương trình:**

Xây dựng giao diện đồ họa để dễ dàng cho việc tra cứu từ.

# Tài liệu tham khảo

[1] http://en.wikipedia.org/wiki/B-tree

[2] http://en.wikipedia.org/wiki/Soundex

[3] http://www.informatik.uni-leipzig.de/~duc/Dict/install.html

[4] http://www.hydrus.org.uk/doc/bt/html/index.html

[5] http://www.math.utah.edu/docs/info/rlman\_toc.html

6. Slide Bài giảng lập trình C nâng cao dành cho chương trình Hedspi - ĐH BKHN