TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------------o0o---------------



**BÀI TẬP LỚN MÔN HỌC**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

Giảng viên hướng dẫn: Phạm Xuân Tích

Sinh viên thực hiện: Trần Công Thành - Lớp CNTT 6 - K61

**Hà Nội, tháng 12 năm 2021**

Mục lục

[I. Đề bài 3](#_Toc89186318)

[II. Phân tích bài toán 4](#_Toc89186319)

[1. Xác định các lớp, các thuộc tính, các phương thức của lớp và mô tả chức năng của từng lớp, từng phương thức. 4](#_Toc89186320)

[a. Lớp **Node** 4](#_Toc89186321)

[b. Lớp **sList(Single Linked List)** 5](#_Toc89186322)

[c. Lớp **hashTable(bảng băm)** 6](#_Toc89186323)

[d. Lớp **anhViet(từ điển)** 7](#_Toc89186324)

[2. Vẽ sơ đồ khối và phân tích thời gian chạy (if, else, for) 8](#_Toc89186325)

[a. **Node <Keys, T> \*insertLast(Keys key, T v)** 8](#_Toc89186326)

[b. **void remove(Node <Keys, T> \*p)** 9](#_Toc89186327)

[c. **Node <Keys, T> \*getNode(Keys k)** 10](#_Toc89186328)

[d. **Node <Keys, T> \*Add(Keys key, T \_val, int (\*hash)(Keys, int))** 11](#_Toc89186329)

[e. **void Remove(Keys key, int (\*hash)(Keys, int))** 12](#_Toc89186330)

[f. **long Count()** 13](#_Toc89186331)

[g. **void Travel(ofstream &ofs)** 14](#_Toc89186331)

[h. **bool editWord(string word, string newMean)** 15](#_Toc89186331)

[i. **void readFile()** 16](#_Toc89186331)

[3. Phân tích thời gian chạy của từng phương thức](#_Toc89186325) 17

[III. Cài đặt các lớp và hàm main bằng C++ 1](#_Toc89186339)9

[IV. Danh sách tài liệu tham khảo 3](#_Toc89186340)3

# Đề bài

1. Xây dựng lớp biểu diễn một bảng băm sử dụng một trong những hàm băm đã học
2. Sử dụng lớp bảng băm để xây dựng lớp biểu diễn từ điển Anh Việt. Mỗi phần tử của từ điển là một cặp (tiếng anh, nghĩa tiếng việt).
   * Với các phương thức:
     + Nạp từ điển từ file vào bảng băm
     + Tìm kiếm bằng phương pháp tìm kiếm trên bảng băm
     + Sửa đổi các từ
     + Thêm từ mới
     + Xóa từ
     + Lưu từ điển vào file
3. Xây dựng chương trình có các chức năng
   * Nạp từ điển từ file (có thể nạp tự động khi chạy chương trình)
   * Tra từ điển (nhập vào một từ, hiển thị nghĩa của nó nếu có trong từ điển)
   * Sửa đổi từ
   * Thêm từ mới
   * Xóa từ
   * Lưu từ điển vào file

# Phân tích bài toán

## Xác định các lớp, các thuộc tính, các phương thức của lớp và mô tả chức năng của từng lớp, từng phương thức.

### Lớp Node ()

Là một **Node** trong các **Node** liên kết với nhau tạo thành **Danh sách liên kết.**

* *Các thuộc tính:*

+, **key**: Là phần tử kiểu mẫu **Keys**, dùng để lưu khóa của đối tượng (Node)

+, **val**: Là phần tử kiểu mẫu **T**, lưu giá trị của đối tượng. (Node)

+, **next**: Con trỏ kiểu **Node <Keys, T> \*** , dùng để trỏ đến **Node** tiếp theo trong danh sách.

* *Các phương thức:*

+, Hàm tạo không đối **Node():** Dùng để khởi tạo Node mới không trỏ đến **Node** nào cả, có **key** = 0, **val** = 0, **next** = 0;

+, Hàm tạo có đối **Node(Keys \_key, T \_val)**: Dùng để khởi tạo **Node** mới không trỏ đến **Node** nào cả: **next** = 0; đồng thời gán giá trị cho **Node** bằng cách gán giá trị các thuộc tính:

**key** = \_key; **val** = \_val;

+, **Keys getKey()**: Dùng để lấy **key** của **Node** hiện tại.

+, **void setKey(Keys)**: Dùng để gán giá trị cho thuộc tính **key** của **Node** hiện tại bằng đối của phương thức.

+, **T getVal()**: Dùng để lấy **val** của **Node** hiện tại.

+, **void setVal(T)**: Dùng để gán giá trị cho thuộc tính **val** của **Node** hiện tại bằng đối của phương thức.

+, **Node <Keys, T> \*getNext()**: Dùng để lấy địa chỉ của **Node** do con trỏ **next** trỏ tới.

+, **void setNext(Node <Keys, T> \*)**: Dùng để gán **Node** do con trỏ **next** trỏ tới bằng đối của phương thức.

### Lớp sList (Danh sách liên kết đơn)

**Single Linked List** là một cấu trúc dữ liệu tuyến tính bao gồm một nhóm các **Node** tạo thành một danh sách. Dùng để lưu trữ **danh sách các đối tượng** (Ví dụ: Danh sách khách hàng, Danh sách sinh viên,… )

* *Các thuộc tính:*

+, **head**: Con trỏ kiểu **Node <Keys, T> \*** , dùng để trỏ đến **Node** đầu tiên trong danh sách.

+, **trail**: Con trỏ kiểu **Node <Keys, T> \***, dùng để trỏ đến **Node** cuối cùng trong danh sách.

+, **sz**: giá trị kiểu **long**, dùng để lưu trữ số lượng **Node** có trong **sList**.

* *Các phương thức:*

+, Hàm tạo không đối **sList()**: Dùng để khởi tạo **sList** rỗng có **head** = **trail** = 0 (null) và **sz** = 0.

+, **long size()**: Trả về kích thước hiện tại của **sList**.

+, **bool empty()**: Kiểm tra xem **sList** có rỗng hay không.

+, **Node <Keys, T> \*insertLast(Keys, T)**: thêm một **Node** vào cuối danh sách, trả lại địa chỉ của **Node** đó.

+, **Node <Keys, T> \*replace(Node <Keys, T> \*, Keys, T)**: thay đổi giá trị thuộc tính **key** và **val** của một **Node,** do con trỏ cho trước trỏ tới, bằng các giá trị mới, trả lại địa chỉ của **Node** đó.

+, **void remove(Node <Keys, T> \*)**: Loại bỏ **Node** do con trỏ cho trước trỏ tới.

+, **Node <Keys, T> \*getNode(Keys)**: trả về địa chỉ của **Node** có **key** bằng giá trị cho trước.

+, **Node <Keys, T> \*getHead(Keys)**: trả về con trỏ **Head;**

### Lớp hashTable (bảng băm)

**Bảng băm** là một cấu trúc dữ liệu dùng để chứa cặp **key/value**. Sử dụng hàm băm như 1 ánh xạ tuyến tính để chèn hoặc tìm kiếm dữ liệu dễ dàng hơn. Trong trường hợp lý tưởng, việc thao tác với các phần tử trong bảng băm sẽ có độ phức tạp O(1).

Ở đây ta sử dụng bảng băm với kĩ thuật : **Open Hashing.**

* *Các thuộc tính:*

+, **N**: giá trị kiểu **long**, dùng để lưu trữ số lượng **sList** có trong **hashTable**.

+, **table**: Con trỏ dùng làm mảng động để lưu trữ các danh sách liên kết.

* *Các phương thức:*

+, Hàm tạo có đối **hashTable(int )**: khởi tạo **hashTable** có số lượng **sList** cho trước.

+, **Node <Keys, T> \*Add(Keys key, T obj, int (\*hash)(Keys, int))**: thêm một **Node** vào trong **hashTable**, trả lại địa chỉ của **Node** đó, nếu tồn **Node** có thuộc tính **key** có giá trị bằng giá trị cho trước thì trả về 0 (null). (với **hàm băm** cho trước)

+, **void Remove(Keys key, int (\*hash)(Keys, int))** : Loại bỏ **Node** có giá trị của thuộc tính **key** bằng với giá trị cho trước trong **hashTable**. (với **hàm băm** cho trước)

+, **Node <Keys, T> \*Find(Keys key, int (\*hash)(Keys, int))** : trả về địa chỉ của **Node** có giá trị của thuộc tính **key** bằng với giá trị cho trước trong **hashTable**. (với **hàm băm** cho trước)

+, **bool Contains(Keys key, int (\*hash)(Keys, int))** : kiểm tra xem có **Node** có giá trị của thuộc tính **key** bằng với giá trị cho trước trong **hashTable** hay không. (với **hàm băm** cho trước)

+, **long Count()**: trả về số **Node** có trong **hashTable**.

+, **void Travel(ofstream &)**: in ra giá trị của tất cả các cặp thuộc tính **key** & **val** của các **Node** trong **hashTable** ra 1 file được trỏ bởi con trỏ file cho trước.

**\* Hàm băm:**

**Hàm băm** được sử dụng để chuyển đổi từ khóa thành chỉ số (giá trị băm) trong mảng lưu trữ các giá trị tìm kiếm.

**Hàm băm** sử dụng trong bài toán được đặt ra: lấy **tổng mã ASCII** của từng kí tự trong chuỗi tiếng Anh, sau đó chia lấy dư cho kích thước của **hashTable**.

int Hash(string key, int n)

{

int sum = 0;

for(int i = 0; i < key.length(); i++) sum += key[i];

return sum % n;

}

### Lớp anhViet

Xây dựng từ class **hashTable**, dùng để thực hiện công việc của 1 từ điển Anh – Việt. Mỗi Node trong **hashTable** của từ điển sẽ là 1 cặp từ Anh - Việt. **Khóa** ở đây là từ tiếng Anh.

* *Các thuộc tính:*

+, **Dic**: Con trỏ kiểu **hashTable <string, string> \*** , **bảng băm** từ điển Anh – Việt

* *Các phương thức:*

+, Hàm tạo không đối **anhViet()**: Dùng để khởi tạo **hashTable** có số **sList** cố định, cụ thể ở đây là 101, đánh chỉ số từ 0 đến 100.

+, **bool addWord(string, string)**: Thêm cặp từ Anh-Việt vào từ điển, nếu đã tồn tại từ tiếng Anh trong từ điển thì trả về **false**, nếu không tồn tại và thêm cặp từ mới thành công thì trả về **true.**

+, **bool editWord(string, string)**: Sửa đổi cặp từ Anh – Việt có sẵn trong từ điển, nếu không tồn tại từ tiếng Anh trong từ điển thì trả về **false**, nếu tồn tại từ tiếng Anh và sửa nghĩa tiếng Việt của nó thành công thì trả về **true**. (Chỉ sửa được nghĩa tiếng Việt)

+, **void removeWord(string)**: Xóa 1 cặp từ Anh – Việt có trong từ điển theo khóa là từ tiếng Anh.

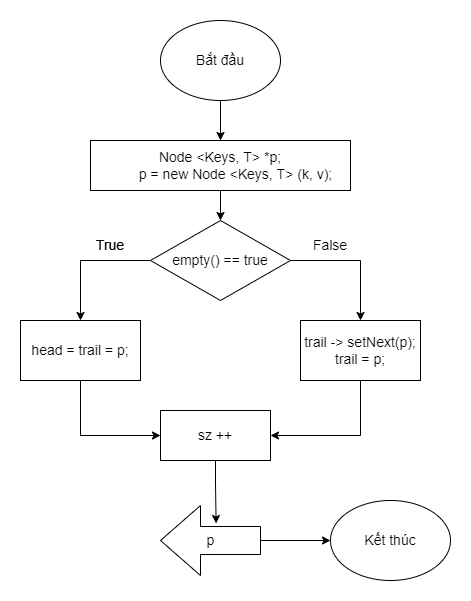
+, **void readFile()**: Nạp file từ điển tự động từ file “**tudien.txt**”.

+, **void exportFile()**: Xuất file từ điển ra file cho trước.

## Vẽ sơ đồ khối và phân tích thời gian chạy.

* **Lớp List:**

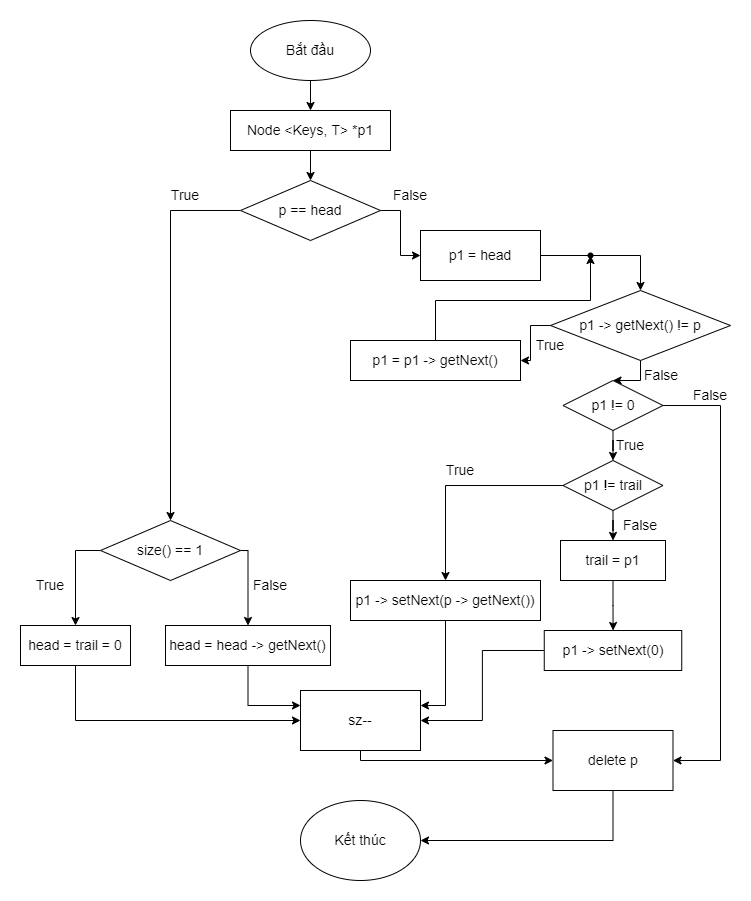
### Node <Keys, T> \*insertLast(Keys key, T v)



**Nhận xét**: Vì ta chỉ kiểm tra điều kiện 1 lần duy nhất, sau đó thêm **Node** vào cuối danh sách với sự hỗ trợ của con trỏ **trail** nên độ phức tạp thuật toán là 1 hằng số.

* **Độ phức tạp thời gian của thuật toán là:** O(1);

### void remove(Node <Keys, T> \*p)



**Nhận xét**: Trong trường hợp xấu nhất, nếu xóa 1 **Node** ở vị trí cuối cùng trong danh sách

thì ta phải duyệt tất cả các phần tử của danh sách từ đầu đến cuối. Vì vậy thuật toán sẽ có độ phức tạp tuyến tính.

* **Độ phức tạp thời gian của thuật toán là:** O(n);

### Node <Keys, T> \*getNode(Keys k)

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCacheContent.Word\getNode.drawio.png

**Nhận xét**: Trong trường hợp xấu nhất, tất cả các , nếu **Node** cần tìm không có trong danh sách thì ta vẫn phải duyệt tất cả các phần tử của danh sách từ đầu đến cuối. Vì vậy thuật toán sẽ có độ phức tạp tuyến tính.

* **Độ phức tạp thời gian của thuật toán là:** O(n);
* **Lớp hashTable:**

### Node <Keys, T> \*Add(Keys key, T \_val, int (\*hash)(Keys, int)

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Add.png

**Nhận xét**: Trong trường hợp xuất nhất. khi thêm 1 **Node** vào **bảng băm,** ta sẽ sử dụng hàm **getNode()** sẽ có độ phức tạp tuyến tính và kiểm tra điều kiện để không thêm các **Node** có **key** trùng lặp, hàm **insertLast()** chỉ có độ phức tạp hàng số, nên thuật toán thêm 1 **Node** vào trong **bảng băm** sẽ có độ phức tạp tuyến tính.

* **Độ phức tạp thời gian của thuật toán là:** O(n);

### void Remove(Keys key, int (\*hash)(Keys, int))

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCacheContent.Word\htbRemove.drawio.png

**Nhận xét**: Trong trường hợp xấu nhất khi xóa 1 **Node** có khóa cho trước từ **bảng băm,** ta sẽ sử dụng hàm **getNode()** có độ phức tạp tuyến tính để tìm **Node** có cùng **key**, hàm **remove()** cũng có độ phức tạp tuyến tính, nên thuật toán khi xóa 1 **Node** khỏi **bảng băm** sẽ có độ phức tạp tuyến tính.

* **Độ phức tạp thời gian của thuật toán là:** O(n);

### long Count()

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCacheContent.Word\count.png

**Nhận xét**: Khi tính tổng số **Node** có trong **bảng băm**, ta cần duyệt qua **N bucket** hay **N sList,** với mỗi 1 địa chỉ, ta sẽ gọi hàm **size()** có độ phức tạp là hằng số, nên thuật toán sẽ có độ phức tạp tuyến tính.

* **Độ phức tạp thời gian của thuật toán là:** O(N);

### void Travel(ofstream &ofs)

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCacheContent.Word\travel.png

**Nhận xét**: để in ra giá trị 2 thuộc tính **key** và **val** của **Node**, dù ta có băm và xếp các **Node** như thế nào thì ta cũng cần duyệt qua **N sList** (N ô địa chỉ), đồng thời cần duyệt qua và in ra tất cả **n** cặp **key** và **val**.

* **Độ phức tạp thời gian của thuật toán là:** O(N + n);
* **Lớp anhViet:**

### bool editWord(string word, string newMean)

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCacheContent.Word\editWord.png

**Nhận xét**: để sửa 1 cặp từcó trong **bảng băm**, ta cần dùng hàm **Find()** để tìm từ tiếng Anh trong từ điển, sau đó nếu có thì mới sửa nghĩa tiếng Việt, trong trường hợp xấu nhất ,hàm **Find()** độ phức tạp tuyến tính, nên thuật toán có độ phức tạp tuyến tính.

* **Độ phức tạp thời gian của thuật toán là:** O(N);

### void readFile()

### C:\Users\Admin\AppData\Local\Microsoft\Windows\INetCacheContent.Word\readFile.png

**Nhận xét**: để thêm **1 cặp từ** vào từ điểncó trong **bảng băm**, ta cần dùng hàm **addWord()** có độ phức tạp thuật toán tuyến tính trong trường hợp xấu nhất. Và ta có **n** cặp từ, vì vậy thuật toán có độ phức tạp bình phương.

* **Độ phức tạp thời gian của thuật toán là:** O(n^2);

## Xác định thời gian chạy của từng phương thức.

### Lớp Node ()

+, **Node():** O(1);

+, **Node(Keys, T)**: O(1);

+, **getKey()**: O(1).

+, **setKey(Keys)**: O(1).

+, **getVal()**: O(1).

+, **setVal()**: O(1).

+, **getNext()**: O(1).

+, **setNext()**: O(1).

### Lớp sList (Danh sách liên kết đơn)

+, **sList()**: O(1).

+, **size()**: O(1).

+, **empty()**: O(1).

+, **insertLast()**: Tốt nhất: O(1) Tệ nhất: O(n).

+, **replace()**: O(1).

+, **remove()**: Tốt nhất: O(1) Tệ nhất: O(n).

+, **getNode()**: Tốt nhất: O(1) Tệ nhất: O(n)

+, **getHead()**: O(1).

### Lớp hashTable (bảng băm)

+, **hashTable()**: O(1).

+, **Add()**: Tốt nhất: O(1) Tệ nhất: O(n).

+, **Remove()**: Tốt nhất: O(1) Tệ nhất: O(n).

+, **Find()**: Tốt nhất: O(1) Tệ nhất: O(n).

+, **Contains()**: Tốt nhất: O(1) Tệ nhất: O(n).

+, **long Count()**: O(N).

+, **Travel()**: O(N+n).

### Lớp anhViet

+, **anhViet()**: O(1).

+, **addWord()**: Tốt nhất: O(1) Tệ nhất: O(n).

+, **editWord()**: Tốt nhất: O(1) Tệ nhất: O(n)..

+, **removeWord()**: Tốt nhất: O(1) Tệ nhất: O(n)..

+, **readFile()**: Tốt nhất: O(n) Tệ nhất: O(n^2).

+, **exportFile()**: O(N + n).

# Cài đặt các lớp và hàm main bằng C++

+, Lớp **Node** được lưu trữ trong thư mục “**node.cpp**”

#ifndef node\_\_cpp

#define node\_\_cpp

#include <iostream>

using namespace std;

template <class Keys, class T>

class Node

{

Keys key;

T val;

Node <Keys, T> \*next;

public:

Node()

{

key = 0;

val = 0;

next = 0;

}

Node(Keys \_key, T \_val)

{

key = \_key;

val = \_val;

next = 0;

}

Keys getKey()

{

return key;

}

void setKey(Keys \_key)

{

key = \_key;

}

Node <Keys, T> \*getNext()

{

return next;

}

void setNext(Node <Keys, T> \*p)

{

next = p;

}

T getVal()

{

return val;

}

void setVal(T \_val)

{

val = \_val;

}

};

#endif

+, Lớp **sList** được lưu trữ trong thư mục “**sList.cpp**”

#ifndef sList\_\_cpp

#define sList\_\_cpp

#include "node.cpp"

#include <iostream>

using namespace std;

template <class Keys, class T>

class sList

{

long sz;

Node <Keys, T> \*head;

Node <Keys, T> \*trail;

public:

sList()

{

head = trail = 0;

sz = 0;

}

long size()

{

return sz;

}

bool empty()

{

return sz == 0;

}

Node <Keys, T> \*insertLast(Keys k, T v)

{

Node <Keys, T> \*p;

p = new Node <Keys, T> (k, v);

if(empty())

{

head = trail = p;

}

else

{

trail -> setNext(p);

trail = p;

}

sz++;

return p;

}

void replace(Node <Keys, T> \*p, Keys k, T v)

{

p -> setKey(k);

p -> setVal(v);

}

void remove(Node <Keys, T> \*p)

{

Node <Keys, T> \*p1;

if(p == head)

{

if(size() == 1) head = trail = 0;

else head = head -> getNext();

sz--;

}

else

{

p1 = head;

while(p1 -> getNext() != p) p1 = p1 -> getNext();

if(p1 != 0)

{

if(p != trail) p1 -> setNext(p -> getNext());

else

{

trail = p1;

p1 -> setNext(0);

}

sz--;

}

}

delete p;

}

Node <Keys, T> \*getNode(Keys k)

{

Node <Keys, T> \*p = head;

while(p != 0 && p -> getKey() != k) p = p -> getNext();

return p;

}

Node <Keys, T> \*getHead()

{

return head;

}

};

#endif

+, Lớp **hashTable** được lưu trữ trong thư mục “**hashTable.cpp**”

#ifndef hashTable\_\_cpp

#define hashTable\_\_cpp

#include"sList.cpp"

#include <iostream>

#include <fstream>

using namespace std;

int Hash(string key, int n)

{

int sum = 0;

for(int i = 0; i < key.length(); i++) sum += key[i];

return sum % n;

}

template <class Keys, class T>

class hashTable

{

int N;

sList <Keys, T> \*table;

public:

hashTable(int n)

{

N = n;

table = new sList <Keys, T> [N + 1];

}

Node <Keys, T> \*Add(Keys key, T obj, int (\*hash)(Keys, int))

{

Node <Keys, T> \*p;

int h = hash(key, N);

p = table[h].getNode(key);

if(p == 0) return table[h].insertLast(key, obj);

else return 0;

}

void Remove(Keys key, int (\*hash)(Keys, int))

{

Node <Keys,T> \*p;

int h = hash(key, N);

p = table[h].getNode(key);

if(p != 0) table[h].remove(p);

}

Node <Keys, T> \*Find(Keys key, int (\*hash)(Keys, int))

{

int h = hash(key, N);

return table[h].getNode(key);

}

bool Contains(Keys key, int (\*hash)(Keys, int))

{

int h = hash(key, N);

return table[h].getNode(key);

}

long Count()

{

long s = 0;

for(int i = 0; i < N; i++) s += table[i].size();

return s;

}

void Travel(string s)

{

ofstream ofs(s);

Node <Keys, T> \*x;

for(int i = 0; i < N; i++)

{

x = table[i].getHead();

while(x != 0)

{

ofs << x -> getKey() << endl << x -> getVal() << endl;

x = x -> getNext();

}

}

ofs.close();

}

};

#endif

+, Lớp **anhViet** được lưu trữ trong thư mục “**anhViet.cpp**”

#ifndef anhViet\_\_cpp

#define anhViet\_cpp

#include <iostream>

#include <fstream>

#include"hashTable.cpp"

using namespace std;

class anhViet

{

hashTable <string, string> \*Dic;

public:

anhViet()

{

Dic = new hashTable <string, string> (101);

}

bool addWord(string word, string mean)

{

return Dic -> Add(word, mean, Hash);

}

bool editWord(string word, string newMean)

{

Node <string, string> \*p;

p = Dic -> Find(word, Hash);

if(p == 0) return 0;

else

{

p -> setVal(newMean);

return 1;

}

}

void removeWord(string word)

{

Dic -> Remove(word, Hash);

}

Node <string, string> \*findWord(string word)

{

return Dic -> Find(word, Hash);

}

void readFile()

{

ifstream ifs("tudien.txt");

string word, mean;

while(!ifs.eof())

{

getline(ifs, word);

getline(ifs, mean);

Dic -> Add(word, mean, Hash);

}

ifs.close();

}

void exportFile(string s)

{

Dic -> Travel(s);

}

};

#endif

+, Menu và chương trình được lưu trữ trong thư mục “**final.cpp**”

#include <bits/stdc++.h>

#include "anhViet.cpp"

#include <conio.h>

using namespace std;

int menu()

{

int n;

cout << "=================MENU================\n";

cout << "1. Nap tu dien tu file\n";

cout << "2. Tra tu dien\n";

cout << "3. Sua doi tu\n";

cout << "4. Them tu moi\n";

cout << "5. Xoa tu\n";

cout << "6. Them tu dien vao file\n";

cout << "7. Ket thuc\n";

cout << "=====================================\n";

cout << "Moi chon: ";

cin >> n;

return n;

}

void demo(anhViet aV)

{

int c;

do

{

system("cls");

c = menu();

system("cls");

switch(c)

{

case 1:

{

cout << "1. Nap tu dien tu file\n";

aV.readFile();

cout << "Da nhap file tu dong thanh cong !!";

break;

}

case 2:

{

cout << "2. Tra tu dien\n";

string word;

cout << "Moi nhap tu tieng Anh can tim: ";

fflush(stdin);

getline(cin, word);

Node <string, string> \*p = aV.findWord(word);

if(p) cout << "Nghia cua tu vua nhap: " << p -> getVal();

else cout << "Tu vua nhap khong co trong tu dien !!";

break;

}

case 3:

{

cout << "3. Sua doi tu\n";

string word, newMean;

cout << "Moi nhap tu tieng Anh: ";

fflush(stdin);

getline(cin, word);

cout << "Moi nhap nghia tieng Viet moi: ";

fflush(stdin);

getline(cin, newMean);

if(aV.editWord(word, newMean)) cout << "Da sua tu thanh cong !!";

else cout << "tu tieng Anh khong co trong tu dien !!";

break;

}

case 4:

{

cout << "4. Them tu moi\n";

string word, mean;

cout << "Moi nhap tu tieng Anh: ";

fflush(stdin);

getline(cin, word);

cout << "Moi nhap nghia tieng Viet: ";

fflush(stdin);

getline(cin, mean);

if(aV.addWord(word, mean)) cout << "Da them vao tu dien thanh cong !!";

else cout << "tu tieng Anh da ton tai trong tu dien !!";

break;

}

case 5:

{

cout << "5. Xoa tu\n";

string word;

cout << "Moi nhap tu tieng Anh: ";

fflush(stdin);

getline(cin, word);

aV.removeWord(word);

cout << "Xoa tu thanh cong !!";

break;

}

case 6:

{

cout << "6. Them tu dien vao file\n";

string s;

cout << "Moi nhap ten file muon ghi tu dien vao: ";

fflush(stdin);

cin >> s;

aV.exportFile(s);

cout << "Da xuat tu dien ra tep !!";

break;

}

case 7:

{

cout << "7. Ket thuc\n";

cout << "Nhan nut bat ki de ket thuc!";

break;

}

}

getch();

}while(c != 7);

}

int main()

{

anhViet aV;

demo(aV);

return 0;

}

# Danh sách tài liệu tham khảo

1. Slide bài giảng môn học Cấu trúc dữ liệu và Giải thuật, *Lecture 7 Linked List.ppt*, Trường Đại Học Giao Thông Vận Tải
2. Slide bài giảng môn học Cấu trúc dữ liệu và Giải thuật, *Lectures 14 Hash tables.ppt*, Trường Đại Học Giao Thông Vận Tải
3. <https://vimentor.com/vi/lesson/ba-ng-bam-hash-table>
4. <https://nguyenvanhieu.vn/bang-bam-hash-tables/>