**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

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



**BÀI TẬP LỚN**

**Môn học: Cấu trúc dữ liệu và giải thuật**

Giảng viên: Hoàng Văn Thông

Sinh viên thực hiện: Vũ Hữu Hào – 201200112

Lớp CNTT 5-K61

**BÀI LÀM**

1. **Giới thiệu đề tài**

Đề 11: Để kiểm tra lỗi chính tả trong các văn bản, người ta thực hiện kiểm tra từng từ của văn bản có trong từ điển của ngôn ngữ đó hay không?. Việc kiểm tra lỗi chính tả được sử dụng trong nhiều ứng dụng như các phần mềm soạn thảo văn bản. Hãy viết một chương trình kiểm tra lỗi chính tả văn bản tiếng anh theo các yêu cầu sau:

- Nạp một từ điển ngôn ngữ tiếng Anh vào một cấu trúc thích hợp từ file

-Viết chương trình nhập vào một đoạn văn, thực hiện kiểm tra xem trong đoạn văn đó có từ nào sai lỗi chính tả hay không?

-Liệt kê các từ sai lỗi chính tả, người dùng kiểm tra những từ mà phần báo sai chính tả nhưng nó không sai thì thêm nó vào từ điển

-Ghi từ điển vào file nếu có bổ sung từ mới.

1. **Phân tích bài toán**

Chương trình của em gồm 4 file chính là HASHTBL.cpp, NODE.cpp, SLIST.cpp, DEMO.cpp và một file input.txt để chứa thư viện từ điển.

* **File NODE.cpp:**
* Có 3 thuộc tinh là key, elem, next.
* Có các phương thức: Hàm tạo không đối, hàm tạo có đối, hàm getElem, getNext, setElem, setNext, getKey, setKey.
* Cụ thể chức năng của từng phương thức:

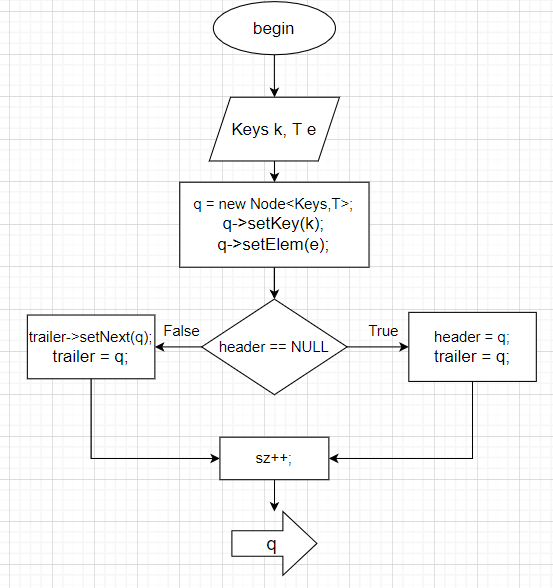
|  |  |
| --- | --- |
| class Node{ |  |
| private: |  |
| Keys key; | // Lưu giá trị khoá |
| T elem; | // Lưu giá trị của node hiện tại |
| Node<Keys,T> \*next; | //Lưu địa chỉ của node tiếp theo |
| public: |  |
| Node(){ | // Hàm tạo không đối |
| next = 0; |  |
| } |  |
| Keys getKey(){ | // Gọi giá trị của khoá |
| return key; |  |
| } |  |
| void setKey(Keys k){ | // Gán giá trị cho khoá |
| key = k; |  |
| } |  |
| Node<Keys,T> \*getNext(){ | //Tra lai dia chi cua doi tương do thanh phan Next tro den |
| return next; |  |
| } |  |
| void setNext(Node<Keys,T> \*p){ | // Dat thanh phan Next tro vao doi tuong la doi cua phuong thuc |
| next = p; |  |
| } |  |
| T getElem(){ |  |
| return elem; | //Tra lai dia chi cua phan tu luu trong Node |
| } |  |
| void setElem(T e){ | //Dat thay the phan tu luu trong Node bang phan tu moi |
| elem = e; |  |
| } |  |
| }; |  |

* Độ phức tạp thuật toán: tất cả phương thức của lớp snode đều có độ phức tạp là O(1)
* **File SLIST.cpp:**
* Có thuộc tính header, trailer, sz.
* Có các phương thức như: hàm tạo có đối, hàm tạo không đối, hàm size, isEmpty, insertLast, getNode.
* Cụ thể các chức năng của từng hàm như sau:

|  |  |
| --- | --- |
| class SingleList{ |  |
| private: |  |
| Node<Keys,T> \*header; | // Lưu địa chỉ của node đầu tiên |
| Node<Keys,T> \*trailer; | // Lưu địa chỉ của node cuối cùng |
| long sz; | // Kích thước của list |
| public: |  |
| SingleList(){ | // Hàm tạo không đối |
| header = NULL; | Độ phức tạp thuật toán : |
| trailer = NULL; | O(1). |
| sz = 0; |  |
| } |  |
| long size(){ | // Phương thức trả về số lượng node trong danh sách |
| return sz; | Độ phức tạp thuật toán :  O(1). |
| } |  |
| int isEmpty(){ | // Hàm kiểm tra list có trống không |
| return sz==0; | Độ phức tạp thuật toán : |
|  | O(1). |
| } |  |

- Node<Keys,T> \*insertLast(Keys k, T e){}: Hàm chèn thêm 1 node vào cuối danh sách.

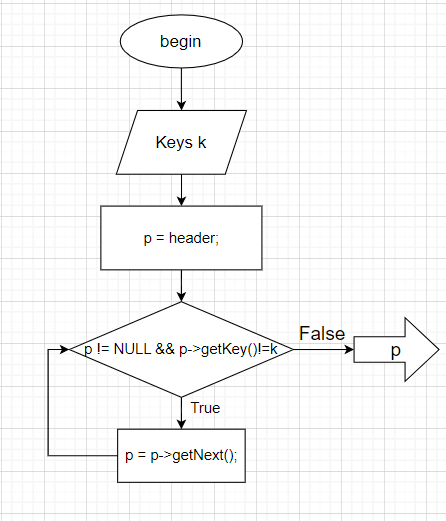
+ Sơ đồ khối của phương thức là:



+ Độ phức tạp thuật toán : O(1)

- Node<Keys,T> \*getNode(Keys k){}: Hàm tìm kiếm tuần tự

+ Sơ đồ khối của phương thức là:



+ Độ phức tạp thuật toán : O(n)

* **File HASHTBL.cpp:**

- Có các thuộc tínhlà: m.

- Có các phương thức như: Hàm tạo có đối, hàm xoá, Add, Find, Contains, Count.

- Cụ thể chức năng từng hàm như sau:

class Hashtable

{

private:

int m; // Kích thước bảng băm

SingleList<Keys,T> \*table;

public:

- Hashtable(int n): Hàm tạo có đối.

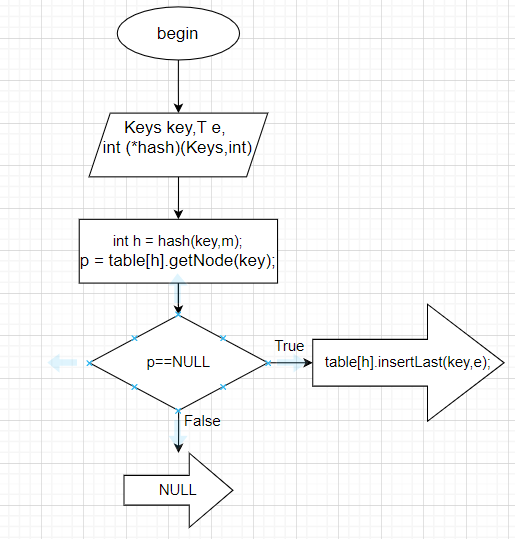
Độ phức tạp thuật toán : O(1)

- ~Hashtable(){}: Hàm huỷ

Độ phức tạp thuật toán : O(1)

- Node<Keys,T> \*Add(Keys key, T obj, int (\*hash)(Keys,int)) : Hàm thêm 1 cái key và obj vào bảng băm

+ Sơ đồ khối của phương thức là:



+ Độ phức tạp thuật toán : O(1)

- Node<Keys,T> \*Find(Keys key, int (\*hash)(Keys,int)): Hàm tìm kiếm phần tử có khóa k nào đó.

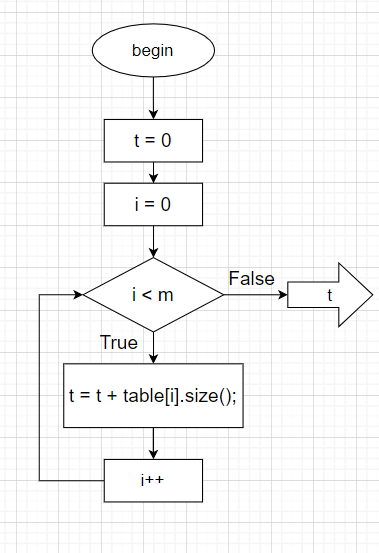
+ Độ phức tạp thuật toán : O(1)

- bool Contains(Keys key, int (\*hash)(Keys,int)): Hàm kiểm tra xem 1 cái key có trong bảng băm chưa.

+ Độ phức tạp thuật toán : O(1)

- int Count(): Hàm đếm toàn bộ bảng có bao nhiêu phần tử

+ Sơ đồ khối của phương thức là:



+ Độ phức tạp thuật toán : O(n)

**3. Cài đặt các lớp và hàm main bằng C++**

File DEMO.cpp gồm có 3 hàm chức năng và 1 hàm main.

* Hàm 1: stringToInt (Hàm tinh tổng mã ASCII của 1 từ )

+ Độ phức tạp thuật toán : O(n).

* Hàm 2: h (Hàm băm để chia các từ vào từng ngăn của bảng băm)

+ Độ phức tạp thuật toán : O(1).

* Hàm 3: tachVanBan ( Hàm để tách văn bản nhập vào thành các từ, rồi lưu vào vector)

+ Độ phức tạp thuật toán : O(n).

* Hàm main:

(1) Nạp file từ điển vào bảng băm.

(2) Nhập 1 đoạn văn bản cần kiểm tra, in ra các từ sai chinh tả nếu có .

(3) Nhập các từ muốn thêm vào từ điển, nhấp chữ “end” để dừng thao tác thêm từ mới vào từ điển .

(4) Nhập số 1 để quay lại bước 2, nhập 2 để dừng chương trình .

**4. Tài liệu tham khảo**

* Tham khảo slide và code phần hash table của thầy Hoàng Văn Thông
* Tham khảo phần nhập xuất file từ trang: <https://codelearn.io/sharing/su-dung-file-sieu-co-ban-voi-cpp>