**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG.TPHCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO BÀI TẬP VỀ NHÀ TUẦN 9**

**APPLICATION OF DICTIONARY**

**using HASH TABLE**

| Giảng viên thực hành |

Lê Tuấn Thu

| Nhóm thực hiện |

Nguyễn Huy Hoàng – 21120458

Nguyễn Ngọc Như Huyền - 21120475

Thành phố Hồ Chí Minh – 2022

* **NHÓM HÀM DATA**
* Là nơi chứa những hằng số, cấu trúc dữ liệu do người lập trình định nghĩa và dùng xuyên suốt chương trình.
* **Khai báo thư viện**

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include <fstream> |
|  | #include <string> |
|  | #include <vector> |
|  | using namespace std; |

* **Thư viện fstream:** sử dụng để đọc/ ghi file.
* **Thư viện string:** sử dụng kiểu dữ liệu string và dùng hàm getline().
* **Thư viện vector:** để sử dụng vector và các hàm vector.
* **Khai báo các biến tĩnh**

|  |  |
| --- | --- |
|  | static const int ENTRIES = 100; // So tu |
|  | static const int DEFAULT\_SIZE = 101; // Kich thuoc mac dinh cua Hash Table |

* Hằng số **ENTRIES** = 100 lưu số từ trong từ điển.
* **DEFAULT\_SIZE** = 101 là kích thước của bảng băm.
* **Khai báo các cấu trúc**

|  |  |
| --- | --- |
|  | /\* Cau truc node cua Hash Table cai dat tu dien \*/ |
|  | struct Node { |
|  | string eng; |
|  | string vi; |
|  | }; |
|  |  |
|  | /\* Cau truc Hash Table \*/ |
|  | struct Hash { |
|  | int itemCount; // Count of dictionary entries |
|  | int hashTableSize; // Table size must be prime |
|  | vector<vector<Node>> hashTable; // Array of pointers to entries |
|  | }; |

* **Struct Node:** lưu từ tiếng Anh và nghĩa của từ (tiếng Việt), kiểu dữ liệu của các từ được lưu trong từ điển là string
* **Struct Hash:** lưu các thông tin cần thiết gồm số từ trong từ điển, kích thước bảng băm, và vector chứa các buckets của HashTable
* **CÁC HÀM KHỞI TẠO VÀ IN BẢNG BĂM**

**Text

Description automatically generated**

* Text

  Description automatically generatedKhởi tạo bảng băm với số từ lưu trữ trong bảng băm, kích thước bảng băm và vector lưu trữ các vị trí trong bảng băm.
* **Text

  Description automatically generatedHÀM BĂM**
* Mã hóa từng ký tự của chuỗi bằng cách sử dụng thuật toán Horner và thêm phép toán mod cho kích thước bảng băm.
* **HÀM SO SÁNH 2 CHUỖI KÝ TỰ KHÔNG PHÂN BIỆT CHỮ IN HOA, CHỮ THƯỜNG**

|  |  |
| --- | --- |
| 1 | // So sanh 2 chuoi ky tu khong phan biet chu hoa, chu thuong |
| 2 | bool compareChar(char& c1, char& c2) |
| 3 | { |
| 4 | return (toupper(c1) == toupper(c2)); |
| 5 | } |
| 6 | bool String\_IgnoreCase(std::string& s1, std::string& s2) { |
| 7 | return((s1.size() == s2.size()) && std::equal(s1.begin(), s1.end(), s2.begin(), &compareChar)); |
| 8 | } |

* Sử dụng hàm **equal()** trong thư viện STL để so sánh bằng cách lặp lại tất cả các phần tử trong cả chuỗi và cho từng ký tự trong range1, kiểm tra xem nó có bằng với tất cả các phần tử tương ứng trong range2 hay không, nếu không thì chuyển đổi cả hai thành chữ hoa và kiểm tra lại. Nếu vẫn không bằng nhau thì trả về false
* **HÀM ĐỌC TỪ ĐIỂN TỪ FILE**

|  |  |
| --- | --- |
|  | /\* Ham doc mot tu \*/ |
|  | void readWord(ifstream& filein, Node& node) { |
|  | getline(filein, node.eng, ':'); |
|  | getline(filein, node.vi); |
|  | } |
|  | /\* Ham doc danh sach tu vung \*/ |
|  | void readDictionary(ifstream& filein, Hash& h) { |
|  | init\_Hash(h); |
|  | while (filein.peek() != EOF) { |
|  | Node node; |
|  | readWord(filein, node); |
|  | add(h, node); |
|  | } |
|  | } |

* **readWord():** Đọc một từ tiếng Anh và nghĩa tiếng Việt của nó (trên 1 dòng), được gọi lại trong hàm readDictionary() để đọc hết danh sách các từ có trong từ điển
* **peek():** Đọc ký tự (không lưu ký tự) đến cuối file thì ngừng.
* Text

  Description automatically generated**HÀM TRẢ VỀ VỊ TRÍ CỦA MỘT TỪ TRONG TỪ ĐIỂN**
* Tính giá trị băm của từ.
* Trả về vị trí của từ xét tại ô có giá trị băm tương ứng nếu tìm thấy. Ngược lại trả về -1.
* **HÀM THÊM MỘT TỪ VÀO TỪ ĐIỂN**

|  |  |
| --- | --- |
|  | void add(Hash& h, Node node, bool& flag) { |
|  | unsigned int hash\_node = 0; |
|  | int nodePos = findWord(h, hash\_node, node.eng); |
|  | if (nodePos == -1) { |
|  | h.hashTable[hash\_node].push\_back(node); |
|  | } |
|  | else |
|  | flag = 1; |
|  | } |

* Tìm vị trí của từ cần thêm (nodePos) bằng hàm findWord với tham số truyền vào là h, hash\_node, node.eng
* Nếu không tìm thấy vị trí của từ này (nghĩa là từ cần thêm chưa có trong từ điển) thì thêm vào sau với hàm **push\_back()**
* Ngược lại, nếu từ cần thêm đã tồn tại trong từ điển thì đặt flag = 1
* **HÀM XÓA MỘT TỪ TRONG TỪ ĐIỂN**

|  |  |
| --- | --- |
|  | void deleteItem(Hash& h, string del) { |
|  | unsigned int hash\_del = 0; |
|  | int delPos = findWord(h, hash\_del, del); |
|  | if (delPos == -1) |
|  | cout << "\nKhong tim thay tu can xoa trong tu dien\n"; |
|  | else { |
|  | h.hashTable[hash\_del].erase(h.hashTable[hash\_del].begin() + delPos); |
|  | cout << "\nDa xoa tu " << del << " ra khoi tu dien\n"; |
|  | } |
|  | } |

* Tìm vị trí của từ cần xóa (delPos) bằng hàm findWord với tham số truyền vào là là h, hash\_del, del
* Nếu không tìm thấy vị trí của từ cần xóa trong từ điển (delPos = -1) thì in ra dòng lệnh thông báo cho người dùng là không tìm thấy từ cần xóa
* Nếu tìm thấy thì thực hiện xóa từ với cú pháp h.hashTable[hash\_del].erase(h.hashTable[hash\_del].begin()+delPos)
* **HÀM CẬP NHẬT NGHĨA MỚI CHO TỪ**

|  |  |
| --- | --- |
|  | /\* Ham cap nhat nghia moi cho tu\*/ |
|  | void updateItem(Hash& h, string word) { |
|  | unsigned int hash\_word = 0; |
|  | string new\_meaning; |
|  | int wordPos = findWord(h, hash\_word, word); |
|  | if (wordPos == -1) |
|  | cout << "\nKhong tim thay tu can cap nhat trong tu dien\n"; |
|  | else { |
|  | cout << "\nNhap nghia moi cho tu: "; |
|  | cin.ignore(); |
|  | getline(cin, new\_meaning); |
|  | h.hashTable[hash\_word][wordPos].vi = new\_meaning; |
|  | cout << "\nDa cap nhat thanh cong nghia cua tu\n"; |
|  | } |
|  | } |

* Tìm vị trí của từ cần cập nhật (wordPos), dùng hàm findWord với tham số truyền  
  vào là h, hash\_word, word
* Nếu không tìm thấy vị trí của từ cần cập nhật (wordPos = -1) thì thông báo cho người dùng không tìm thấy từ cần cập nhật
* Nếu tìm thấy vị trí của từ cần cập nhật thì thực hiện cập nhật nghĩa tiếng Việt mới cho từ bằng cách cho h.hashTable[hash\_word][wordPos].vi = new\_meaning
* **HÀM MENU ĐỂ CHẠY CHƯƠNG TRÌNH**
* **Text

  Description automatically generated**Hàm menu chính được gọi trong hàm main để chạy chương trình