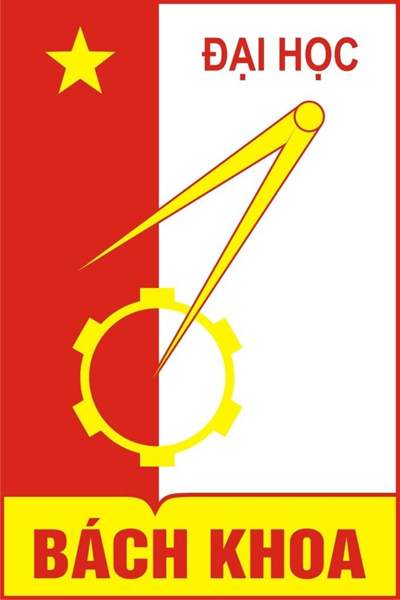
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN – ĐIỆN TỬ**



**BÁO CÁO**

**BÀI TẬP LỚN SỐ 5**

**Đề tài: Xây dựng từ điển từ viết tắt cho người dùng**

**Giảng viên hướng dẫn: TS Đỗ Thị Ngọc Diệp**

**Nhóm 4**

|  |  |
| --- | --- |
| **Sinh viên 1:** | **Phạm Gia Minh 20203741** |
| **Sinh viên 2:** | **Đặng Hữu Công Hiếu 20203417** |
| **Sinh viên 3:** | **Nguyễn Đình Duy 20202357** |

*Hà Nội, ngày 14 tháng 7 năm 2022*

# **Bảng danh sách nhóm:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Họ và tên | Mã số sinh viên | Email | Công việc | Mức độ hoàn thành |
| 1 | Phạm Gia Minh | 20203741 | minh.pg203741@sis.hust.edu.vn | Nội dung báo cáo | 100% |
|  |  |  |  | Tìm lỗi phần mềm | 100% |
|  |  |  |  | Tạo dữ liệu thử nghiệm | 100% |
| 2 | Đặng Hữu Công Hiếu | 20203417 | hieu.dhc203417@sis.hust.edu.vn | Lập nhóm | 100% |
|  |  |  |  | Chọn chủ đề | 100% |
|  |  |  |  | Đưa ra ý tưởng và thiết kế phần mềm | 100% |
|  |  |  |  | Nhận xét và chỉnh sửa, bổ sung, hoàn thiện báo cáo | 100% |
|  |  |  |  | Thiết kế cấu trúc dữ liệu, giải thuật và các chức năng của phần mềm | 100% |
| 3 | Nguyễn Đình Duy | 20202357 | duy.nd202357@sis.hust.edu.vn | Lập nhóm | 100% |
|  |  |  |  | Chọn chủ để | 100% |
|  |  |  |  | Lên lịch trình | 100% |
|  |  |  |  | Chỉnh sửa, bổ sung và hoàn thiện báo cáo | 100% |
|  |  |  |  | Triển khai, chỉnh sửa, tìm và vá lỗi phần mềm | 100% |

Mục lục

[**Bảng danh sách nhóm:** 2](#_Toc108789194)

[**1.** **Giới thiệu bài toán:** 5](#_Toc108789195)

[**Từ điển viết tắt:** 5](#_Toc108789196)

[**Các chức năng:** 5](#_Toc108789197)

[**2.** **Mô tả giải thuật sử dụng:** 6](#_Toc108789198)

[**3.** **Cấu trúc dữ liệu :** 7](#_Toc108789199)

[**Danh sách liên kết đôi:** 7](#_Toc108789200)

[**Thông tin của từ điển:** 7](#_Toc108789201)

[**Cấu trúc dữ liệu của từ điển:** 7](#_Toc108789202)

[**Cấu trúc lưu trữ:** 8](#_Toc108789203)

[**Định nghĩa cấu trúc struct :** 8](#_Toc108789204)

[**Dữ liệu mẫu hiển thị:** 8](#_Toc108789205)

[**4.** **Thiết kế giải thuật :** 9](#_Toc108789206)

[**4.1 Đọc file và nhập vào dữ liệu:** 9](#_Toc108789207)

[**4.2** **Thuật toán thêm vào:** 11](#_Toc108789208)

[**a)** **Thiết kế sơ bộ:** 11](#_Toc108789209)

[**b)** **Thiết kế chi tiết:** 11](#_Toc108789210)

[**4.3** **Thuật toán sửa:** 14](#_Toc108789211)

[**a)** **Thiết kế sơ bộ :** 14](#_Toc108789212)

[**b)** **Thiết kế chi tiết:** 15](#_Toc108789213)

[**4.4** **Thuật toán xóa:** 18](#_Toc108789214)

[**a)** **Thiết kế sơ bộ:** 18](#_Toc108789215)

[**b)** **Thiết kế chi tiết:** 18](#_Toc108789216)

[**4.5** **Thuật toán sắp xếp:** 20](#_Toc108789217)

[**4.5.1: Thuật toán InsertionSort:** 20](#_Toc108789218)

[**4.5.2: Thuật toán Quick Sort:** 21](#_Toc108789219)

[**4.6** **. Thuật toán tìm kiếm:** 22](#_Toc108789220)

[**a)** **Thiết kế sơ bộ:** 22](#_Toc108789221)

[**b)** **Thiết kế chi tiết:** 22](#_Toc108789222)

[**5.** **Phân tích tính đúng đắn của giải thuật.** 25](#_Toc108789223)

[**5.1. Phân tích tính đúng đắn của GT :** 25](#_Toc108789224)

[**5.2.** **Xác định độ phức tạp của GT đề xuất :** 25](#_Toc108789225)

[**6.** **Mô tả triển khai trên ngôn ngữ lập trình cụ thể:** 26](#_Toc108789226)

[**7.** **Tự đánh giá kết quả, các điểm hạn chế:** 28](#_Toc108789227)

[**Lời cảm ơn** 29](#_Toc108789228)

1. **Giới thiệu bài toán:**

## **Từ điển viết tắt:**

Viết tắt là một dạng rút gọn cách viết của một [từ](https://vi.wikipedia.org/wiki/T%E1%BB%AB) hoặc cụm từ. Thông thường, nó bao gồm một hoặc nhiều chữ cái lấy từ chính từ ngữ được viết tắt. Ví dụ, chính chữ *viết tắt* có thể được viết tắt thành "vt".

Viết tắt thường được sử dụng khi câu từ khi viết đầy đủ bị cho là quá dài, hoặc chỗ trống cho việc viết đầy đủ (trên giấy, bảng hiệu) bị thiếu.

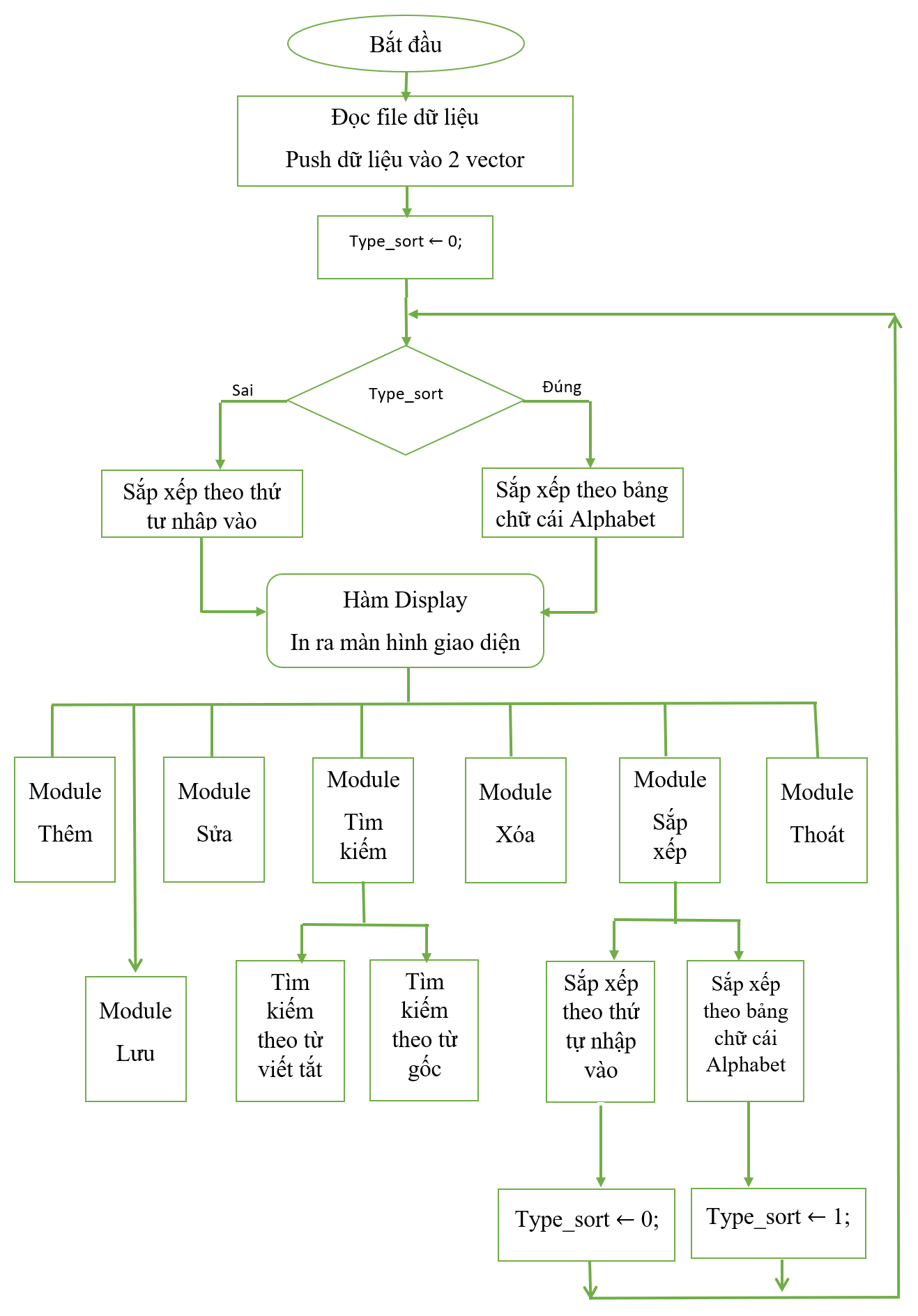
Từ điển viết tắt cho phép xây dựng một từ điển các từ ngữ viết tắt để tra cứu.

Xây dựng một từ điển từ viết tắt cho gồm số thứ tự nhập vào (stt); từ viết tắt (tvt); từ gốc (tg).

## **Các chức năng:**

* Cho phép thêm, sửa, xóa các từ viết tắt cùng dạng viết đầy đủ của nó.
* Cho phép tìm kiếm một từ viết tắt, hiển thị từ viết đầy đủ của nó.
* Cho phép tìm kiếm từ theo dạng viết đầy đủ, hiển thị từ viết tắt của nó.
* Sắp xếp và hiển thị theo thứ tự nhập vào hoặc theo thứ tự alphabet.

1. **Mô tả giải thuật sử dụng:**

****

1. **Cấu trúc dữ liệu :**

**Danh sách liên kết đôi:**

1. Định nghĩa:

Là một tập hợp các **Node** động được phân bố động, được sắp xếp sao cho mỗi Node chứa:

* + Một dữ liệu(data) hoặc nhiều dữ liệu (data có thể là int, string,…)
  + Một con trỏ (**Next)** để trỏ tới phần tử kế tiếp của danh sách liên kết đó, nếu con trỏ trỏ tới **NULL** thì đấy là phần tử cuối cùng của danh sách.
  + Một con trỏ (**Pre**) để trỏ tới phần tử trước của danh sách liên kết đó, nếu con trỏ trỏ tới **NULL** thì đó là phần tử đầu tiên của danh sách.

Graphical user interface

Description automatically generatedGraphical user interface, diagram

Description automatically generated

**Thông tin của từ điển:**

* stt : Số thứ tự nhập vào.
* tvt : Từ viết tắt, chiều dài khoảng từ 0 đến 15 ký tự.
* tg : Từ gốc, chiều dài khoảng từ 0 đến 50 ký tự.

## **Cấu trúc dữ liệu của từ điển:**

* Cấu trúc tuyến tính :
  + Vector
* Cấu trúc phi tuyến:
  + Các module như thêm, sửa, xóa, tìm kiếm, sắp xếp.
* Gồm 2 struct :
  + Một struct để lưu danh sách theo thứ tự thêm vào.
  + Một struct để lưu danh sách theo thứ tự alphabet.
* stt: Số nguyên không âm ( stt>=0)
* tvt: Chuỗi tối đa 20 ký tự.
* tg : Chuỗi tối đa 50 ký tự.
* Các thao tác trên cấu trúc dữ liệu:
* Duyệt phần tử
* Thêm
* Sắp xếp
* Chèn mới
* Tìm kiếm

## **Cấu trúc lưu trữ:**

* Là cấu trúc lưu trữ trong: Nằm trong bộ nhớ trong.
* Là loại cấu trúc lưu trữ động: kích thước dữ liệu có thể thay đổi khi chạy chương trình.

## **Định nghĩa cấu trúc struct :**

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Dữ liệu mẫu hiển thị:**

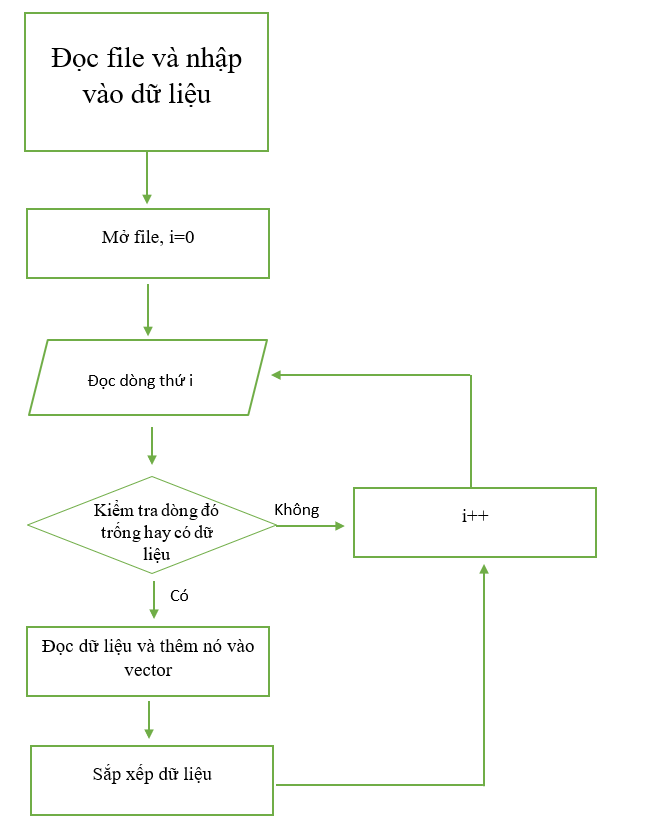
Table

Description automatically generated

1. **Thiết kế giải thuật :**

**4.1 Đọc file và nhập vào dữ liệu:**

**a) Thiết kế sơ bộ:**



1. **Thiết kế chi tiết:**

Bước đầu tiên, mở file, khai báo các biến cần thiết và đọc toàn bộ dữ liệu có trong file.

Text

Description automatically generated

Bước tiếp theo, đọc dữ liệu theo từng dòng và thêm vào vector.

Ảnh có chứa văn bản

Mô tả được tạo tự động

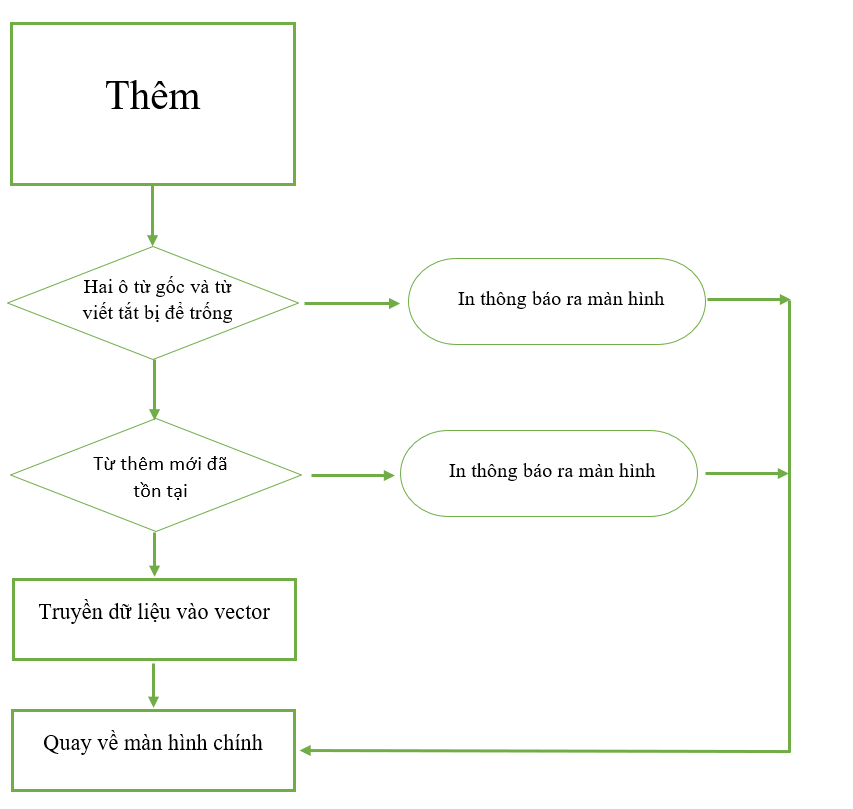
Bước cuối cùng, đóng file và thực hiện thuật toán sắp xếp.

Text

Description automatically generated

* 1. **Thuật toán thêm vào:**

1. **Thiết kế sơ bộ:**

****

1. **Thiết kế chi tiết:**

Kiểm tra xem từ nhập vào có phải là từ rỗng hay không, nếu không phải là từ rỗng thì thông báo ra màn hình. Ảnh có chứa văn bản

Mô tả được tạo tự động

Nếu không thì mới xét xem từ nhập vào đã trùng từ gốc hoặc từ viết tắt hay chưa.

Tìm kiếm tuần tự theo vector lưu theo số thứ tự.Duyệt tất cả các phần tử của vector, đại diện là vector DATABASE\_123

Ảnh có chứa văn bản

Mô tả được tạo tự độngNếu tìm thấy từ trùng lặp thì cho dừng vòng chạy và in ra các thông báo:Ảnh có chứa văn bản

Mô tả được tạo tự động

Khi kiểm tra hết các vector mà không có từ trùng lặp thì thuật toán sẽ bắt đầu thêm từ mới vào.

Lấy giữ liệu trong 2 ô textbox trên giao diện.

Cấp phát động 2 con trỏ với độ dài tương ứng để lưu dữ liệu của từ mới:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Tạo 2 biến là temp\_123 với kiểu DATA\_123 và temp\_abc với kiểu DATA\_abc để push vào 2 vector tương ứng  Thêm vào vector xếp theo số thứ tự trước sau đó thêm vào sắp xếp alphabet:

Nếu là kiểu DATABASE\_123 thì push ngay vào cuối.



Nếu là kiểu DATABASE\_abc thì push như sau:

Duyệt từ đầu đến cuối của vector, đến khi từ viết tắt của từ mới bé hơn từ viết tắt thứ i trong danh sách thì thêm ngay vào vị trí i đó.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Biến p = -1 khi không có từ viết tắt trong danh sách lớn hơn từ viết tắt thêm vào.

Nếu cả vector không có từ viết tắt lớn hơn từ viết tắt mới thì thêm vào cuối vector:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Xóa dữ liệu con trỏ:



Sau khi xóa giữ liệu con trỏ, in ra màn hình đã thêm thành công.

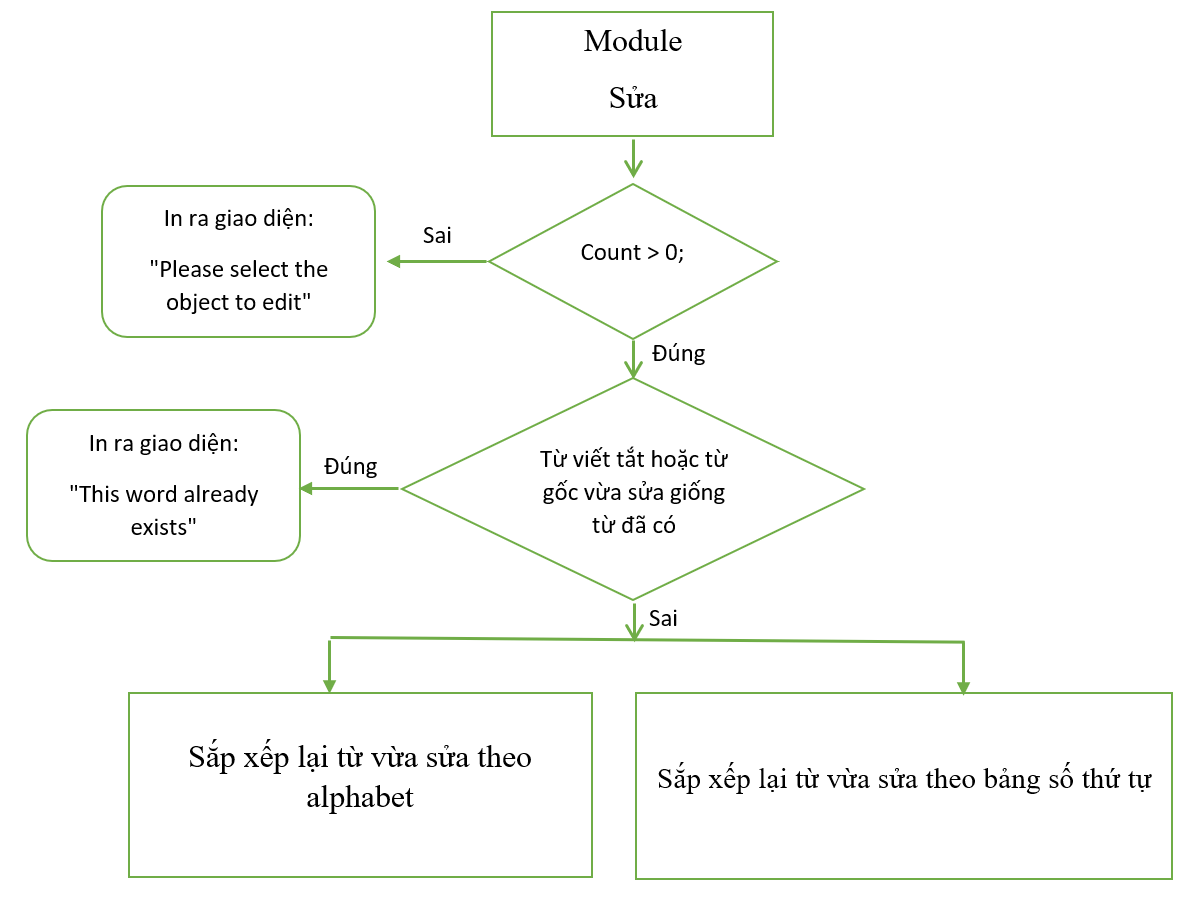
Nếu type\_sort == True thì ra màn hình theo sắp xếp alphabet, nếu không thì sẽ in ra theo số thứ tự đã được thêm vào:

Ảnh có chứa văn bản

Mô tả được tạo tự động

* 1. **Thuật toán sửa:**

1. **Thiết kế sơ bộ :**

****

1. **Thiết kế chi tiết:**

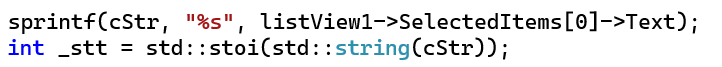
Kiểm tra xem đã ấn chọn từ muốn sửa hay chưa:

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

Biến lưu thứ tự của từ sửa: char cStr[10] = { 0 };

Chuyển vị trí đối tượng thành kiểu int để xác định vị trí :



Tạo con trỏ để lưu giá trị sửa được nhập vào:

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

Kiểm tra tuần tự xem từ mới vừa sửa vào có trùng từ viết tắt hoặc từ gốc đã có hay không :

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

Nếu chọn và sửa ở vector DATABASE\_123 thì sau khi chuyển sang vector DATABASE\_abc không biết từ đó là từ nào, vì 2 vector không có tính liên kết với nhau, nên do đó phải chia thành 2 trường hợp, đó là:

* Chọn và sửa khi đang ở chế độ sắp xếp theo số thứ tự(type\_sort = 0)
* Chọn và sửa khi đang ở chế độ sắp xếp theo alphabet(type\_sort = 1)

***Với type\_sort = 0:*** if (type\_sort == 0)

Truyền giá trị từ giao diện vào vector DATABASE\_123:



Lần lượt so sánh số thứ tự của từ muốn sửa với số thứ tự của vector xếp theo alphabet, nếu giống nhau thì ta sẽ xóa từ gốc và từ viết tắt giống của vector xếp theo số thứ tự :

Text

Description automatically generated

Duyệt từ đầu đến cuối của vector xếp theo alphabet, đến khi từ viết tắt của từ mới bé hơn từ viết tắt thứ i trong danh sách thì thêm ngay vào vị trí i đó.

Text

Description automatically generated

Biến p = -1 khi không có từ viết tắt trong danh sách lớn hơn từ viết tắt thêm vào.

Nếu cả vector không có từ viết tắt lớn hơn từ viết tắt mới thì thêm vào cuối vector:

Text

Description automatically generated with low confidence

Xóa dữ liệu con trỏ:

Ảnh có chứa văn bản

Mô tả được tạo tự động

In ra giao diện màn hình:

Ảnh có chứa văn bản

Mô tả được tạo tự động

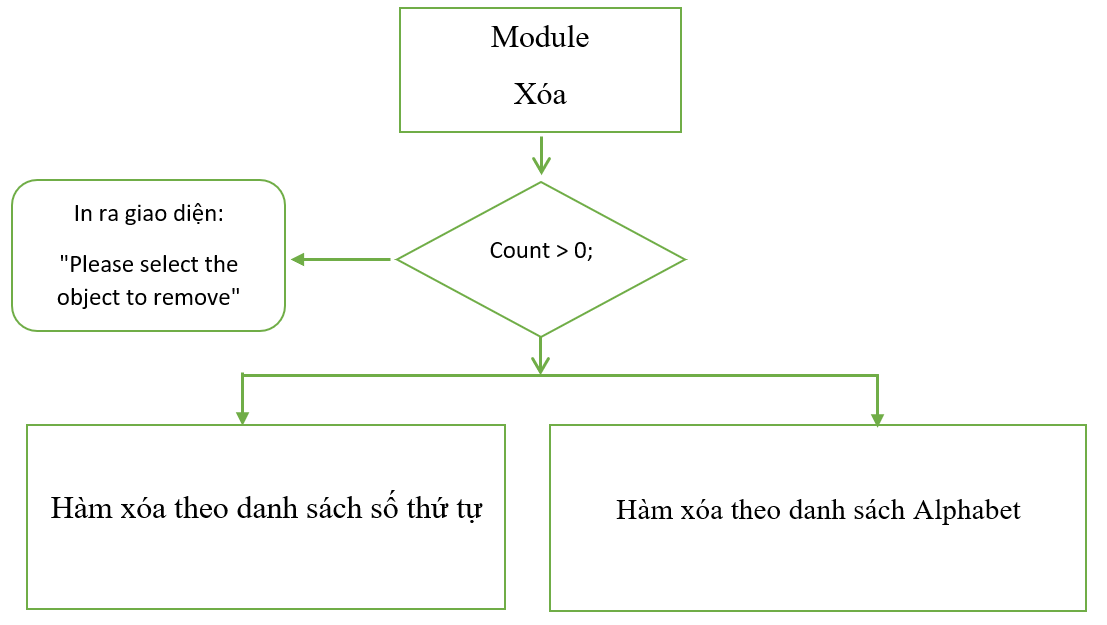
***Với type\_sort = 1:***

Text

Description automatically generated

* 1. **Thuật toán xóa:**

1. **Thiết kế sơ bộ:**



1. **Thiết kế chi tiết:**

Biến lưu thứ tự của từ sửa: char cStr[4] = { 0 };

Chuyển vị trí đối tượng thành kiểu int để xác định vị trí :



Tương tự như thuật toán sửa,nếu xóa ở vector DATABASE\_123 thì sau khi chuyển sang vector DATABASE\_abc không biết từ đó là từ nào, vì 2 vector không có tính liên kết với nhau, nên do đó phải chia thành 2 trường hợp, đó là:

* Chọn và sửa khi đang ở chế độ xóa theo số thứ tự (type\_sort = 0)
* Chọn và sửa khi đang ở chế độ xóa theo alphabet ( type\_sort = 1)

***Với type\_sort = 0 :*** if (type\_sort == 0)

Giảm mỗi số thứ tự ở sau phần tử ở vectorlưu theo số thứ tự được xóa một đơn vị:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Xóa phần tử được chọn: DATABASE\_123.erase(DATABASE\_123.begin() + \_stt - 1);

Lần lượt so sánh số thứ tự của từ muốn sửa với số thứ tự của vector xếp theo alphabet, nếu giống nhau thì ta sẽ xóa từ gốc và từ viết tắt giống của vector xếp theo số thứ tự :

Ảnh có chứa văn bản

Mô tả được tạo tự động

Giảm mỗi số thứ tự ở sau phần tử ở vector lưu theo alphabet được xóa một đơn vị: Ảnh có chứa văn bản

Mô tả được tạo tự động

***Tương tự với type\_sort = 1, ta có:***

Text

Description automatically generated

* 1. **Thuật toán sắp xếp:**

Ở đây, do thuật toán sắp xếp theo thứ tự nhập vào đã được sắp xếp ngay khi nhập nên chỉ cần xây dựng thuật toán sắp xếp theo alphabet.

**4.5.1: Thuật toán InsertionSort:**

**a. Thiết kế sơ bộ:**

Bước 1: truyền vào hàm giá trị m = 0 và n = kích thước vector, gán giá trị

min = a[0].

Bước 2: Tìm phần tử a[min] nhỏ nhất trong dãy hiện hành  từ  a[i] đến a[n-1]

* Nếu a[i] < min thì đổi vị trí của a[min] và a[i]
* Nếu i < N thì tiếp tục chạy, i= i+1;

Lặp lại bước 2

**b. Thiết kế chi tiết:**

Hàm sắp xếp chèn:

Text, letter

Description automatically generated

Hàm đổi vị trí :

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

**4.5.2: Thuật toán Quick Sort:**

**a) Thiết kế sơ bộ:**

* Truyền vào vector giá trị kích thước của vecto
* Chọn phần tử chính giữa làm pivot để chia đôi.
* Cho giá trị chạy từ điểm trái (nhỏ nhất của Partition) có giá trị là i; đến điểm phải (lớn nhất của Partition) có giá trị là j
* Nếu điểm trái >= điểm phải thì dừng lại và return
* Nếu i < j thì:
  + + Nếu phần tử bên trái nhỏ hơn pivot thì bỏ qua, i ++;
  + + Nếu phần tử bên phải nhỏ hơn pivot thì bỏ qua; j--
  + **+** Nếu i < j thì đổi chỗ i và j, sau đó i++ và j--
* Đệ quy: truyền vào hàm Partition(trái, j);
* Đệ quy: truyền vào hàm Partition(phải, j);

**b) Thiết kế chi tiết:**

**Hàm Partition:**

Ảnh có chứa văn bản

Mô tả được tạo tự động

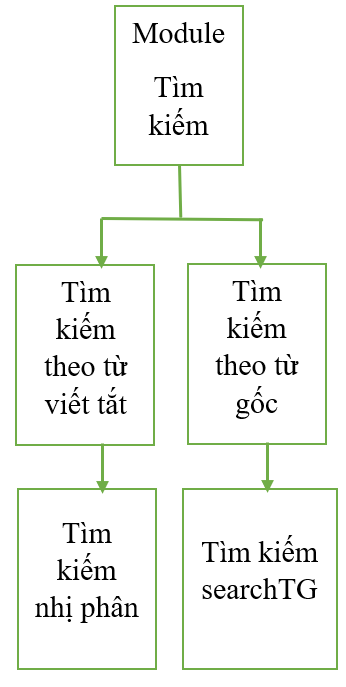
**Hàm hoán đổi:**

Ảnh có chứa văn bản

Mô tả được tạo tự động

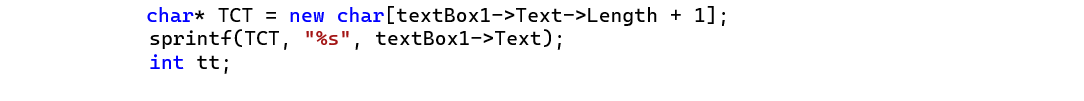
* 1. **. Thuật toán tìm kiếm:**

1. **Thiết kế sơ bộ:**

****

1. **Thiết kế chi tiết:**

Tạo con trỏ lưu từ cần tìm kiếm:



* **Module thuật toán tìm kiếm theo từ viết tắt:**

***Giải thuật tìm kiếm nhị phân:***

int BinarySearchTVT(vector<DATA\_abc> database, string word) {

return BSearchTVT(database, word, 0, database.size() - 1);

}

**Hàm BSearch** :

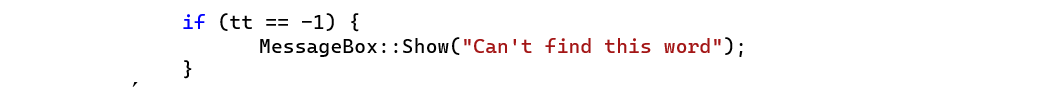
Xét đoạn mảng a[m…n] cần tìm kiếm phần tử tvt. Ta so sánh tvt với phần tử ở vị trí giữa của mảng (k= (m + n)/2). Nếu:

* Nếu phần tử a[k] = tvt. Kết luận và thoát chương trình.
* Nếu a[k] < tvt. Chỉ thực hiện tìm kiếm trên đoạn arr[k+1…n].
* Nếu a[k] > tvt. Chỉ thực hiện tìm kiếm trên đoạn arr[m…k-1].

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

Sau đó, nếu không tìm được vị trí, giá trị của tt vẫn bằng -1, khi đó, in ra giao diện:



Nếu tìm được giá trị, in ra màn hình theo:

**Với type\_sort = 1:**

In đậm từ tìm kiếm theo alphabet:

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Với type\_sort = 0:**

In đậm từ tìm kiếm theo số thứ tự

Ảnh có chứa văn bản

Mô tả được tạo tự động

* **Module thuật toán tìm kiếm theo từ gốc:**

***Sử dụng giải thuật tìm kiếm tuần tự:***

Ảnh có chứa văn bản

Mô tả được tạo tự động

Sau đó, nếu không tìm được vị trí, giá trị của tt vẫn bằng -1, khi đó, in ra giao diện: 

Nếu tìm được giá trị, in ra màn hình theo:

* Với type\_sort = 0: ( in đậm từ tìm kiếm theo số thứ tự):

Graphical user interface, text

Description automatically generated

* Với type\_sort = 1: ( in đậm từ tìm kiếm theo alphabet):

Text

Description automatically generated

1. **Phân tích tính đúng đắn của giải thuật.**

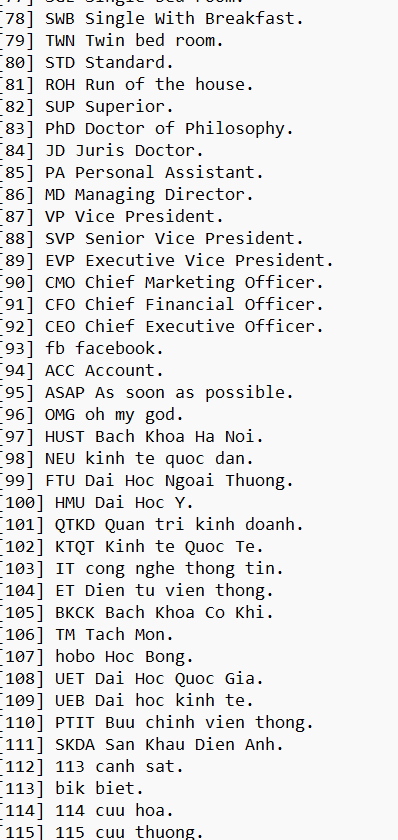
**5.1. Phân tích tính đúng đắn của GT :**

Qua nhiều lần chạy thử nghiệm, chúng em thấy chương trình hoạt động khá ổn định, thực hiện được các chức năng như năng như yêu cầu của bài toán đưa ra, giao diện thân thiện và dễ nhìn.

Việc chạy chương trình không có lỗi gì xảy ra.

File dữ liệu thử lên đến hơn 100 cụm từ viết tắt, gồm các từ khóa đặc biệt, từ nhiều kí tự, các chữ số...

File dữ liệu thử :



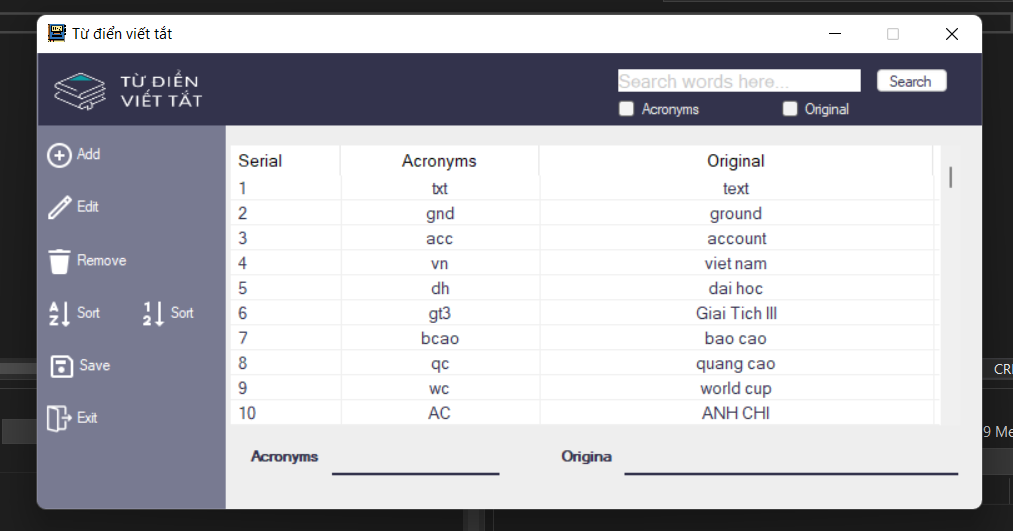
**5.2.** **Xác định độ phức tạp của GT đề xuất :**

* Độ phức tạp của thuật toán đọc file O(n)
* Độ phức tạp của module thêm, sửa, xóa là O(n)
* Độ phức tạp của hàm in ra màn hình giao diện có độ phức tạp là O(n)
* Thuật toán sắp xếp (Selection Sort) là một thuật toán khá đơn giản có độ phức tạp là O(n2) .
* Thuật toán sắp xếp nhanh (Quick Sort) là một thuật toán có độ phức tạp trong trường hợp xấu là O(n2) và trong trường hợp trung bình là O(nlogn)
* **Độ phức tạp của thuật toán tìm kiếm nhị phân là: O = log2(n)**
* **Độ phức tạp của thuật toán tìm kiếm tuần tự là: O = log2(n)**

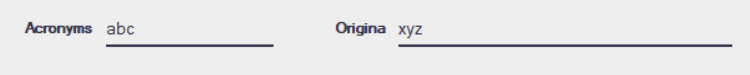
1. **Mô tả triển khai trên ngôn ngữ lập trình cụ thể:**

Khi chạy, tạo ra 1 giao diện có nhiều module để lựa chọn :

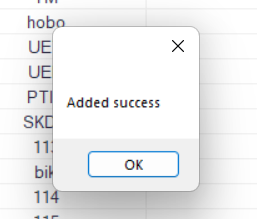
* **Module add:**

****

Khi nhập từ viết tắt và từ gốc vào đây :

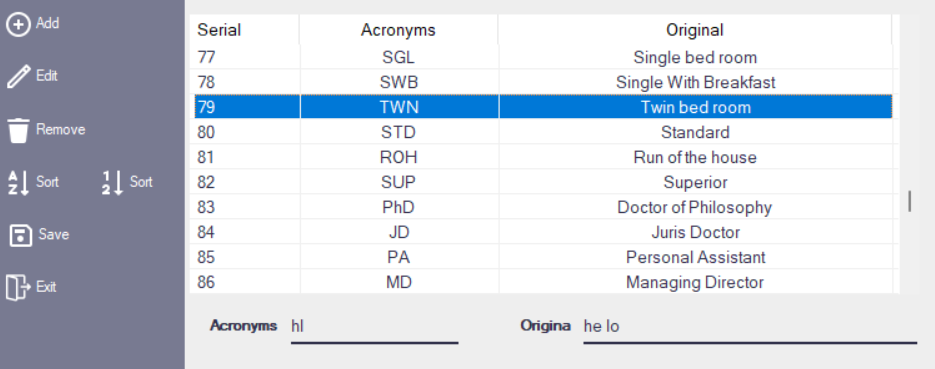


Và ấn vào “Add” sẽ cho ra kết quả như sau:

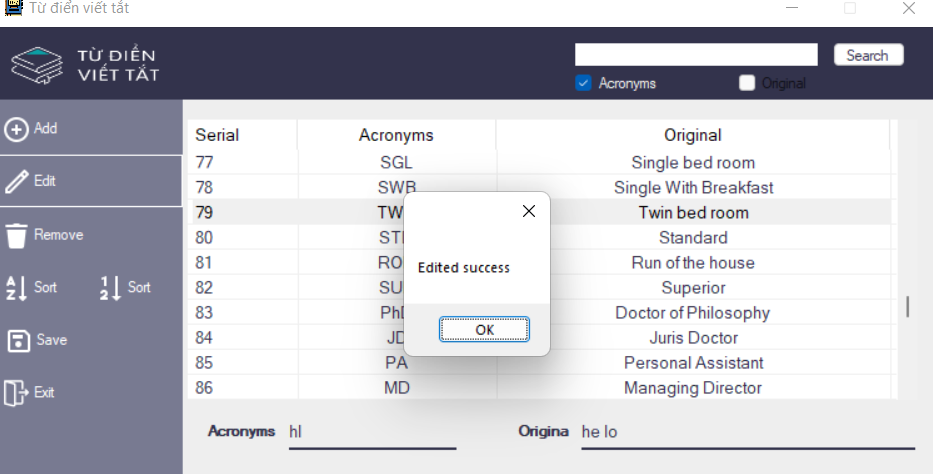


* **Module Edit:**

Khi ấn vào từ cần edit, sau đó nhập lại từ cần edit như hình dưới đây :



Ấn vào “Edit” thành công sẽ ra kết quả như sau:



* **Module sắp xếp:**

Khi chọn vào sắp xếp theo alphabet, ấn vào biểu tượng : 

Kết quả khi sắp xếp theo alphabet:



Khi chọn vào sắp xếp theo số thứ tự, ấn vào biểu tượng : 

Kết quả khi sắp xếp theo số thứ tự:



1. **Tự đánh giá kết quả, các điểm hạn chế:**

Về kết quả, sau nhiều lần chạy thử nghiệm trên chính máy tính cá nhân và một số máy tính của bạn bè, chúng em đã hạn chế đến mức tối đa các lỗi có thể xuất hiện trên app của mình, đồng thời chạy chính xác trên tập dữ liệu mẫu.

Về mặt hạn chế, nhóm chúng em tồn tại duy nhất một vài điểm nhỏ về mặt triển khai bài toán. Thứ nhất, do tính năng của một số lớp có sẵn, giải pháp tối ưu nhóm chúng em tìm được bắt buộc phải tạo ra 2 vector trong giải thuật sắp xếp, dẫn đến tốn bộ nhớ hơn khi chạy. Thứ hai, do giới hạn về mặt công cụ, bắt buộc các chức năng của các buttom phải nằm trong file header mà chương trình đã tạo, dẫn đến file header của chúng em có tổng số dòng code hơn 1000 dòng.

# **Lời cảm ơn**

Trên đây là báo cáo của nhóm 4 về đề tài Từ điển viết tắt. Do lần đầu được làm việc với nhau, cũng do nhiều yếu tố khác dẫn đến bản báo cáo của nhóm chúng em có thể vẫn còn nhiễu thiếu sót và lỗi. Mong cô và các bạn có thể đóng góp thêm ý kiến để nhóm chúng em có thể rút kinh nghiệm và hoàn thiện bản thân mình hơn trong tương lai.

Cuối cùng, chúng em xin phép gửi lời cảm ơn đến cô Đỗ Thị Ngọc Diệp, người đã giảng dạy và hướng dẫn chúng em trong suốt thời gian qua. Một lần nữa cảm ơn cô rất nhiều.