**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

****

**BÁO CÁO ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

***Đề tài:***

**ỨNG DỤNG BẢNG BĂM XÂY DỰNG TỪ ĐIỂN ANH - VIỆT**

***Giáo viên hướng dẫn:* Trần Công Tú**

***Lớp:* Sáng thứ 7**

***SVTH :* NGUYỄN ĐÌNH PHÚ**

***MSSV :* 18110175**

***SVTH :* NGUYỄN NGỌC KHANG**

***MSSV :* 18110133**

***TP. Hồ Chí Minh, tháng 11, năm 2019***

Mục lục

[A. DANH MỤC CÁC HÌNH 2](#_Toc27148916)

[B. PHẦN MỞ ĐẦU 3](#_Toc27148917)

[I. Lý do chọn đề tài 3](#_Toc27148918)

[II. Mục tiêu 3](#_Toc27148919)

[C. NỘI DUNG 4](#_Toc27148920)

[I. Mô tả project 4](#_Toc27148921)

[1. Mục đích sử dụng 4](#_Toc27148922)

[2. Cấu trúc dữ liệu & giải thuật được dùng 4](#_Toc27148923)

[II. Mô tả quá trình làm 4](#_Toc27148924)

[1. Thiết kế giao diện 4](#_Toc27148925)

[2. Thiết kế cấu trúc dữ liệu & giải thuật 7](#_Toc27148926)

[3. Cài đặt và kiểm thử 12](#_Toc27148927)

[III. Mô tả phân công công việc 14](#_Toc27148928)

[IV. Kết luận 15](#_Toc27148929)

[1. Đánh giá mức độ hoàn thành 15](#_Toc27148930)

[2. Khó khăn khi xây dựng chương trình 15](#_Toc27148931)

[3. Ưu khuyết điểm của từ điển 15](#_Toc27148932)

[4. Hướng phát triển 16](#_Toc27148933)

[D. TÀI LIỆU THAM KHẢO 17](#_Toc27148934)

# DANH MỤC CÁC HÌNH

[Hình 1: Giao diện chính của chương trình 5](#_Toc27148935)

[Hình 2: Thông báo thêm từ 6](#_Toc27148936)

[Hình 3: Giao diện thêm từ 6](#_Toc27148937)

[Hình 4: Thông báo hoàn tất thêm từ 7](#_Toc27148938)

[Hình 5: Hàm Băm 8](#_Toc27148939)

[Hình 6: Cập nhật file dữ liệu mới vào từ điển 10](#_Toc27148940)

[Hình 7: Cài đặt bảng băm và xử lý va chạm 11](#_Toc27148941)

[Hình 8: Tìm kiếm trên bảng băm 12](#_Toc27148942)

# PHẦN MỞ ĐẦU

## Lý do chọn đề tài

Trong thời kì hội nhập và phát triển hiện nay của nước ta cùng với sự phát triển mạnh mẽ của các nước trên thế giới. Ngoài những khía cạnh về kinh tế, thương mại, cạnh tranh hàng hóa,… Ngôn ngữ cũng là một khía cạnh quan trọng khi nói về sự hội nhập. Tiếng Anh là một ngôn ngữ rất phổ biến hiện nay, được sử dụng trên đa số các quốc gia trên thế giới. Tuy nhiên để học một ngôn ngữ mới không phải là dễ. Trong quá trình học ngôn ngữ mới chắc hẳn sẽ có nhiều khó khăn, vì vậy một công cụ có ích sẽ ít nhiều hỗ trợ cho việc học trở nên dễ dàng hơn. Và từ điển chắc chắn là một công cụ không thể thiếu với việc học ngôn ngữ. Với sự phổ biến của Tiếng Anh trên thế giới, các thành viên trong nhóm đã xây dựng một chương trình từ điển Anh - Việt.

## Mục tiêu

Phần quan trọng nhất đối với một ứng dụng từ điển không phải là khả năng hoạt động của ứng dụng đó, mà lại chính là cơ sở dữ liệu. Việc xây dựng cơ sở dữ liệu cho từ điển phải đảm bảo được khả năng truy cập nhanh cho ứng dụng bởi dữ liệu của từ điển thường khá lớn, lên tới hàng chục đến hàng trăm nghìn từ. Đặc biệt chương trình có sử dụng bảng băm, là một cấu trúc dữ liệu tốt cho việc xây dựng và tìm kiếm. Bảng băm tỏ ra rất vượt trội vì tốc độ tìm kiếm vô cùng nhanh nếu xây dựng được một hàm băm tốt.

Chương trình có một thiết kế đơn giản, dễ sử dụng, người dùng có thể dễ dàng tra cứu từ vựng Tiếng Anh, người dùng còn có thể thêm từ vựng vào dữ liệu của từ điển theo ý muốn. Đồng thời chương trình có thể giúp cho người dùng có được một từ điển cho riêng mình nhằm phục vụ các mục đích cụ thể khác nhau.

# NỘI DUNG

## Mô tả project

### Mục đích sử dụng

Hiện nay các từ điển online và phần mềm từ điển tuy đã rất nhiều, nhưng để có một dữ liệu từ điển cho riêng mình thì các từ điển có sẵn không thể làm được. Chẳng hạn từ ngữ chuyên ngành hay một vấn đề gì đó hoặc dùng nội bộ thì các từ điển có sẵn hay từ điển online gần như không thể đáp ứng được.

Hơn nữa các ví dụ và chi tiết trong từ không thể mô tả đầy đủ và chi tiết về chuyên ngành hay vấn đề mình đang sử dụng. Thêm vào đó từ điển cho phép người dùng thêm từ mới, do đó giúp người dùng tự phát triển dữ liệu từ điển cho riêng mình.

### Cấu trúc dữ liệu & giải thuật được dùng

Cấu trúc dữ liệu & giải thuật được sử dụng là bảng băm.

Đặc điểm của bảng băm là từ từ khóa cần tìm dùng mã băm để băm đến danh sách chứa vị trí nghĩa, khả năng này là nhảy trực tiếp đến nghĩa, vô cùng nhanh, nhưng rất phụ thuộc vào hàm băm và dữ liệu nhập vào, tốc độ tìm kiếm phụ thuộc nhiều vào bảng băm.

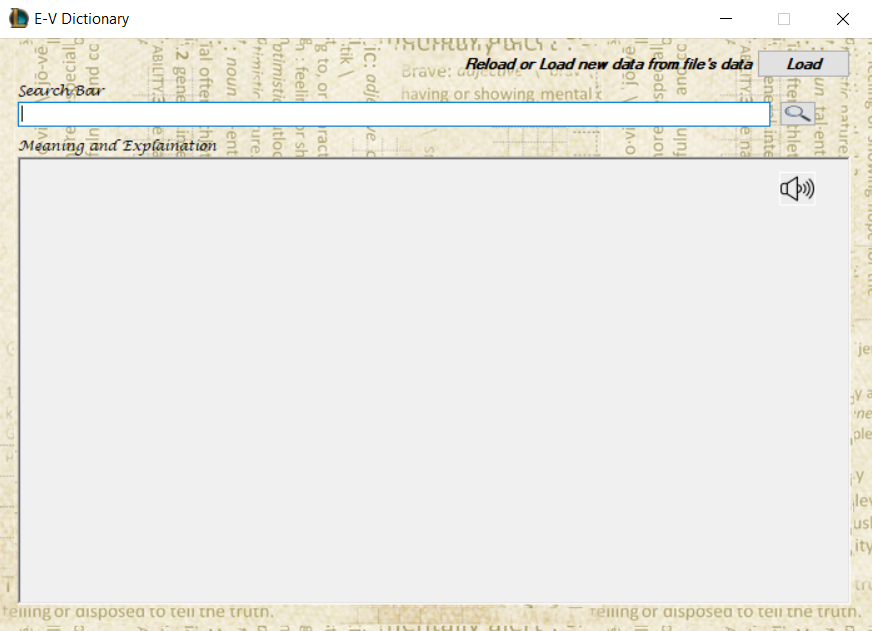
So với các cách dùng cơ sở dữ liệu khác như Access, xml, SQL,… có độ tùy biến không cao, tốc độ truy vấn dữ liệu thấp thì bảng băm tỏ ra vượt trội hơn.

## Mô tả quá trình làm

### Thiết kế giao diện

#### Giao diện chính của từ điển

* Khi người dùng mở chương trình sẽ xuất hiện form sau:

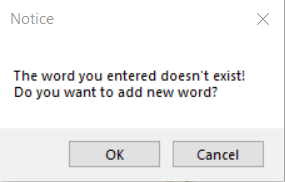


Hình 1: Giao diện chính của chương trình

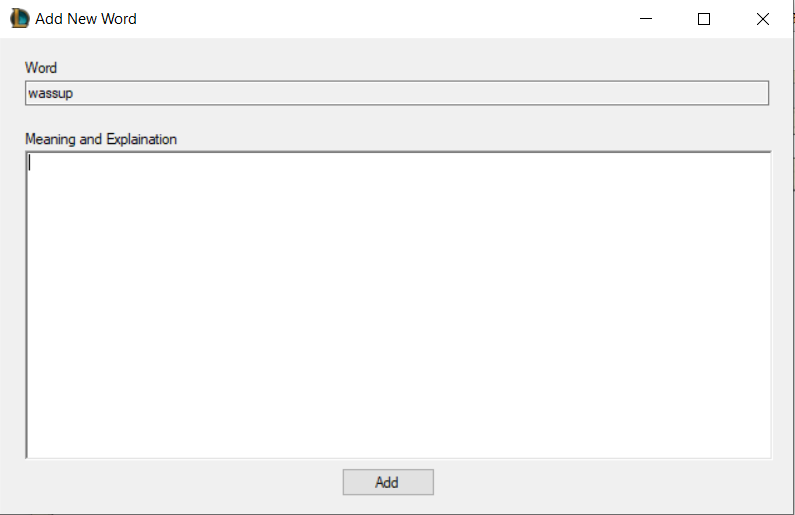
* Giao diện chính của từ điển gồm các thành phần:
* TextBox *‘textBox1’* dùng để nhập từ cần tra.
* RichTextBox *‘richTextBox1’* dùng để hiện nghĩa từ, phiên âm và giải thích.
* Label *‘label1’* text là *‘Search Bar’*.
* Label *‘label2’* text là *‘Meaning and Explaination’*.
* Label *‘label3’* text là *‘Reload or Load new data from file’s data’*.
* Button *‘FindBt’* có tác dụng như phím Enter, sau khi người dùng nhập từ cần tra và nhấn vào, nghĩa từ và phần giải thích sẽ hiện ra ở *‘richTextbox1’*.
* Button *‘SpeakBt’* có tác dụng phát âm từ mà người dùng vừa nhập.
* Button *‘LoadBt’* có tác dụng cập nhật hoặc cập nhật lại dữ liệu trong từ điển từ file dữ liệu có sẵn.

#### Giao diện thêm từ mới

* Khi từ người dùng nhập vào không có trong dữ liệu từ điển thì chương trình sẽ hiện lên thông báo:

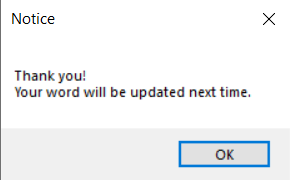


Hình 2: Thông báo thêm từ

* Nếu người dùng nhấn OK giao diện thêm từ sẽ hiện ra:

Hình 3: Giao diện thêm từ

* Giao diện thêm từ gồm các thành phần:
* TextBox *‘addNewKey’* sẽ hiển thị từ mà người dùng cần thêm, ở đây sẽ để mặc định là từ mà người dùng vừa tra nên sẽ không thay đổi được.
* RichTextBox *‘addMeaningBox’* sẽ để người dùng nhập nghĩa từ, phiên âm và giải thích theo ý của người dùng.
* Label *‘label1’* có text là *‘Word’*.
* Label *‘label2’* có text là *‘Meaning and Explaination’*.

\* Button *‘AddBt’* có text là *‘Add’* có tác dụng thêm từ sau khi người dùng hoàn thành nhập nghĩa và nhấn vào. Sau khi hoàn tất thêm từ, chương trình sẽ hiện lên thông báo:

Hình 4: Thông báo hoàn tất thêm từ

### Thiết kế cấu trúc dữ liệu & giải thuật

#### Dữ liệu chính được lưu trong cấu trúc

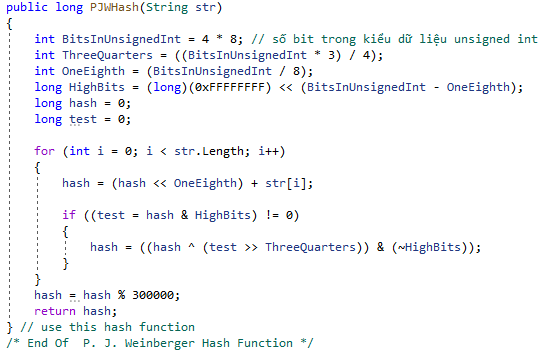
Dữ liệu chính được lưu trong cấu trúc là một mảng chứa các gói từ - nghĩa có kiểu dữ liệu là DictionaryData (một kiểu dữ liệu trừu tượng do người lập trình định nghĩa bởi class), kiểu dữ liệu này chứa hai elements là từ khóa (key) và nghĩa (meaning) – bao gồm chi tiết của từ như ví dụ, loại từ, cách dùng,… hai elements này có kiểu dữ liệu là string.

#### Các thao tác chính trên cấu trúc dữ liệu

Thao tác chính là load dữ liệu từ file dữ liệu lên RAM, để tăng tốc độ load, ta sẽ dùng hàm băm để băm trước key ra mã băm và khi load dữ liệu ta chỉ cần đưa mã băm của từ đó vào index của bảng băm.

#### Giải thuật chính đã sử dụng

* ***Băm:***



Hình 5: Hàm Băm

* Hàm băm dùng để chuyển từ khóa tìm kiếm (string) thành mã băm (số nguyên).

Hàm băm ảnh hưởng trực tiếp đến độ hiệu quả của bảng băm vì nếu một hàm băm tốt sẽ giảm thiểu va chạm, từ đó làm tăng tốc độ tìm kiếm. Vì đối với mỗi phần tử bị va chạm ta sẽ phải thêm phần xử lí tìm kiếm cho nó, càng va chạm nhiều, số lần xử lí này sẽ tăng lên do đó làm tăng thời gian xử lí tìm kiếm.

Có nhiều cách xử lí va chạm nhưng trong đề tài này ta chỉ sử dụng ***Kĩ thuật phân tách – Separate chaining***: là một kỹ thuật xử lý và chạm phổ biến nhất. Nó thường được cài đặt với danh sách liên kết. Để lưu giữ một phần tử trong bảng băm, ta phải thêm nó vào một danh sách liên kết ứng với chỉ mục của nó. Nếu có sự va chạm xảy ra, các phần tử đó sẽ nằm cùng trong 1 danh sách liên kết. Tuy nhiên trong đồ án này ta sẽ tận dụng điểm mạnh của ngôn ngữ mang tính hướng đối tượng là C#, ta sẽ tạo ra đối tượng trong đó chứa *“Key”*, *“Meaning”* và một danh sách chứa các phần tử va chạm, danh sách này thay cho danh sách liên kết nói trên. Điều này sẽ dễ dàng hơn cho việc xử lí dữ liệu trong danh sách bởi sự hỗ trợ tốt từ ngôn ngữ C#.

Tuy nhiên cách xử lí va chạm ***Separate chaining*** tốn khá nhiều bộ nhớ do phụ thuộc vào giới hạn trên của giá trị mã băm và vẫn còn những ô nhớ không được dùng tới. Vì vậy để đạt được hiệu quả tối đa thì ta cần có một hàm băm tốt để va chạm là tối thiểu, từ đó danh sách va chạm sẽ nhỏ dẫn tới tìm kiếm nhanh, đồng thời các ô nhớ trống sẽ được sử dụng, từ đó giảm phung phí bộ nhớ.

Trong trường hợp lí tưởng, ta sẽ có số lần xử lí tìm kiếm là 1 (hay thời gian tìm kiếm là O(1)), điều này chỉ xảy ra khi không có sự va chạm đối với phần tử cần tìm kiếm. Trường hợp tệ nhất có thời gian tìm kiếm là O(n) khi tất cả các phần tử của bảng bẳm đều va chạm với nhau và phần tử cần tìm kiếm nằm ở cuối danh sách va chạm.

Hàm băm được sử dụng trong đề tài được tham khảo từ hàm băm của ***Peter Jay Weinberger***, hàm băm này có thể coi là khá tốt khi có thể giảm được tỉ lệ va chạm so với một số cách thông thường được dùng như:

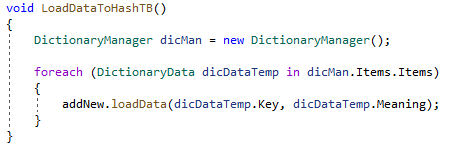
* Cộng tất cả giá trị ASCII của mỗi kí tự trong chuỗi key lại với nhau (có thể chia lấy dư với n để giới hạn giá trị mã băm).
* Cộng tất cả các giá trị ASCII của mỗi kí tự trong chuỗi key sau khi đã nhân với số chỉ vị trí của kí tự tương ứng rồi chia lấy dư với n để giới hạn giá trị mã băm.

Những cách tương tự kể trên rất không hiệu quả chẳng hạn nếu cộng lại thì các chuỗi có cùng các kí tự với thứ tự khác nhau sẽ giống nhau hay chuỗi có các kí tự với mã ASCII lớn nhưng độ dài chuỗi ngắn sẽ va chạm với chuỗi dài với các kí tự có mã ASCII nhỏ,…

Với hàm băm được dùng ở đây ta sẽ xử lí trên bit. Đầu tiên mã băm sẽ được giới hạn ở giá trị của kiểu int (số nguyên) là 32 bit, sau đó để giới hạn giá trị tối đa cho mã băm phù hợp với độ lớn dữ liệu, ta sẽ chia lấy dư với 300 000. Chi tiết như sau:

Highbit là 4 bit đầu tiên của kiểu int (ở đây được lưu trong kiểu long). Ban đầu ta có mã băm hash = 0. Sau đó với mỗi kí tự của chuỗi được duyệt từ trái sang phải ta dịch hash sang trái 4 bit rồi cộng với mã ASCII của kí tự đó.

Nếu số bit đã dùng lớn hơn 28 thì ta lấy 4 bit đầu (tương ứng với kiểu int) gán trước 4 bit 0. Ta được 8 bit trong đó 4 bit đầu là 4 bit đã lấy và 4 bit sau là 0 sau đó xor với hash, điều này sẽ giảm thiểu sự trùng lặp bit nếu 4 bit đầu và 4 bit 5, 6, 7, 8 từ phải qua trái giống nhau. Sau đó ta sẽ and bit với Highbit được đảo bit để lấy 28 bit cuối (tránh tràn bit khỏi giới hạn của kiểu dữ liệu int).



Hình 6: Cập nhật file dữ liệu mới vào từ điển

Hàm *LoadDataToHashTB* thực hiện đưa file dữ liệu chỉ gồm các từ khóa và nghĩa chưa được băm vào từ điển. Trong đó hàm *loadData* là hàm thêm một phần tử vào bảng băm (được giải thích ở phần “Thêm một phần tử vào file dữ liệu”). Khi đó file dữ liệu được đưa vào sẽ được lưu ở một file *data.xml* đã bao gồm mã băm trong mỗi phần tử.

* ***Thêm một phần tử vào file dữ liệu***

Hàm *loadData* thực hiện chức năng đưa một phần tử chưa có mã băm từ file *dataF1.xml* vào file *data.xml* và chứa mã băm được tạo ra bởi hàm băm *PJWHash*.

*loadData* được gọi khi đưa dữ liệu mới *(dataF1.xml)* vào file dữ liệu chứa mã băm của từ điển (*data.xml)* và khi thêm một phần tử vào từ điển nếu phần tử được tìm kiếm không có trong dữ liệu của từ điển

* ***Cài đặt bảng băm và xử lý va chạm***

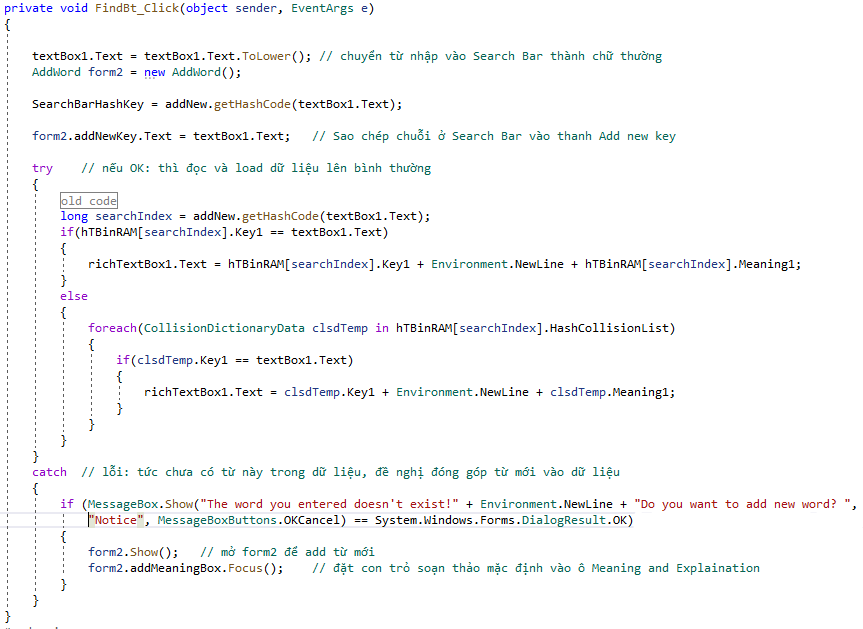


Hình 7: Cài đặt bảng băm và xử lý va chạm

Khi khởi chạy chương trình, hàm *loadTBtoRAM* sẽ được gọi và dữ liệu từ điển (đã bao gồm mã băm trong mỗi phần tử) từ file *data.xml* sẽ được đưa vào mảng băm trên bộ nhớ trong (RAM). Mỗi phẩn tử sẽ được đưa vào vị trí *index* của mảng băm *(MtrDictionaryDataInRAM[index]).* Trong đó mỗi phần tử sẽ bao gồm từ khóa (key) và nghĩa (meaning), và danh sách các phần tử “va chạm” với phần tử này *(List<CollisionDictionaryData>)*.

Các phần tử “va chạm” sẽ là các *CollisionDictionayData* gồm từ khóa (key) và nghĩa (meaning) có cùng mã băm với phần tử *MtrDictionaryDataInRAM[index]* nói trên.

* ***Tìm kiếm trên bảng băm***



Hình 8: Tìm kiếm trên bảng băm

Giải thuật tìm kiếm bằng bảng băm là khi từ khóa tìm kiếm (key) được nhập vào, hàm băm sẽ chuyển đổi từ khóa đó (string) thành mã băm (số nguyên) lưu trong *searchIndex,* là vị trí của phần tử trong mảng băm, nhờ đó sẽ trực tiếp tìm đến phần tử có key nói trên, hoặc danh sách các phân tử “va chạm” cùng mã băm đó. Nếu không có phần tử nào chứa từ khóa trên, phần mềm sẽ đề nghị người dùng đóng góp nghĩa của từ này và thêm từ mới vào từ điển (lưu trong file *data.xml*).

Nhờ phương pháp tìm kiếm này, trong trường hợp tốt nhất ta sẽ có thời gian tìm kiếm là O(1), khi đó va chạm trên phân tử này không xảy ra. Ngược lại thời gian tìm kiếm sẽ tỉ lệ với số phần tử va chạm.

### Cài đặt và kiểm thử

#### Các trường hợp kiểm thử tiêu biểu

* **Trường hợp 1:** Dữ liệu gồm những từ không va chạm, tìm kiếm thử để kiểm tra việc xây dựng bảng băm và tìm kiếm trên bảng băm có hoạt động hay không.

**Ví dụ:** Ta có các từ ***abashment*** sau khi băm có mã băm là ***61972*** và từ ***abask*** có mã băm là ***185179***, ***abatable - 194293, abate - 185189***. Các từ này không va chạm nên sẽ lưu lần lượt ở các ô 61972, 185179, 194293, 185189 của mảng băm

* **Trường hợp 2:** Dữ liệu có các phần tử va chạm, tìm kiếm các phần tử va chạm với nhau.

**Ví dụ:** ***abalone - 67269, cupule - 67269, needle-lace - 67269*** đều có mã băm giống nhau nên từ đầu tiên được đưa vào mã băm sẽ là phần tử chính, khi tìm đến ô 67269 của mảng băm khi trỏ đến *“Key”* sẽ là giá trị ***abalone*** còn các từ còn lại sẽ nằm trong danh sách va chạm, để tìm các từ bị va chạm ta phải trỏ đến phần danh sách của kiểu dữ liệu để tìm kiếm.

* **Trường hợp 3:** Tìm kiếm một phần tử chưa có trong dữ liệu từ điển, kiểm tra “*thêm phần tử vào từ điển”* sau đó khởi động lại, tìm kiếm từ mới được thêm, 2 trường hợp có thể xảy ra là từ này không va chạm và có va chạm, trường hợp không va chạm từ này sẽ được đặt ở một vị trí mới trong mảng băm, ngược lại sẽ nằm trong danh sách các từ mà nó va chạm. Kiểm tra file dữ liệu *data.xml*.

**Ví dụ:** ***abashment*** ***–*** ***61972***, ***abask*** ***-*** ***185179***, ***abatable - 194293, abate – 185189, abalone - 67269, cupule - 67269, needle-lace – 67269*** là các từ đã có trong từ điển, lúc này thử tìm kiếm từ ***hello*** thì sẽ được thông báo là chưa có từ này, đề nghị thêm từ vào từ điển, nếu người dùng thêm từ thì lúc này từ sẽ có mã băm là ***58927*** không va chạm với các từ đã có trên nên được đặt ở ô thứ 58927 của mảng băm.

* **Trường hợp 4:** Đưa vào dữ liệu mới, sau khi băm và lưu trên file *data.xml*. Kiểm tra file *data.xml* xem dữ liệu đã được băm và lưu có đúng đắn hay không. Lúc này dữ liệu sẽ được lưu dạng như ở các trường hợp trên. Đối với các từ đã tồn tại trong dữ liệu của từ điển thì sẽ bị bỏ qua.

#### Các trường hợp kiểm thử sai và cách khắc phục

* **Trường hợp 1:** Không tìm thấy từ nếu nó nằm trong danh sách va chạm, gây ra “*yêu cầu thêm từ mới vào từ điển”*:

**Ví dụ: *abalone - 67269, cupule - 67269, needle-lace – 67269*** được đưa vào file *data.xml* (chứa mã băm) nhưng các từ ***cupule***, ***needle-lace*** không được đưa vào danh sách va chạm mà nằm riêng ra, nên ghi chạy chương trình chỉ có từ ***needle-lace*** là từ cuối cùng có mã băm này được đưa vào mảng băm (do các từ trước đó tuy được đưa vào nhưng đã bị gán lại khi có cùng mã băm), từ đó thiếu sót dữ liệu.

**Nguyên nhân:** xử lí va chạm bị lỗi, các từ bị va chạm không được thêm vào, nên không có các từ này trong dữ liệu từ điển.

**Khắc phục:** Tạo ra một danh sách “*SubdictionaryData”* bao gồm các từ bị va chạm, khi đó chương trình sẽ đưa phần tử đầu tiên (chưa va chạm) vào vị trí *Index* của mảng băm, còn các từ trong danh sách va chạm sẽ được đưa vào đúng danh sách va chạm của phần tử trên.

* **Trường hợp 2:** Nạp file dữ liệu mới nhưng trong file *data.xml*, các từ đã tồn tại trong dữ liệu từ điển nếu lại được nạp vào thì nằm trong danh sách va chạm của chính từ đó.

**Ví dụ:** các từ đã tồn tại được coi như va chạm và nằm trong danh sách va chạm mới được tạo ở trường hợp trên.

**Nguyên nhân:** Các từ được nạp vào và bị trùng lặp với dữ liệu của từ điển chưa bị loại bỏ mà vẫn được thêm vào như một từ bị va chạm mã băm với từ đã có.

**Khắc phục:** Các từ được nạp vào dữ liệu từ điển sẽ thêm bước kiểm tra xem có trùng lặp hay không, nếu không thì sẽ thêm vào, ngược lại sẽ bỏ qua từ đó.

## Mô tả phân công công việc



## Kết luận

### Đánh giá mức độ hoàn thành

Trong đồ án này nhóm thực hiện đã tìm hiểu được kiến thức về bảng băm, cách tổ chức dữ liệu trên bảng băm và các thao tác trên bảng băm như tìm kiếm, thêm dữ liệu vào bảng băm. Áp dụng vào sản phẩm từ điển để thực hiện chức năng tra từ, thêm từ vào từ điển. Nhìn chung các thuật toán được áp dụng một cách linh hoạt để có thể hoàn thành ứng dụng từ điển này.

Tuy nhiên sản phầm này vẫn chỉ là một ứng dụng để mô phỏng cấu trúc dữ liệu và các giải thuật của bảng băm, do đó chưa thể hiệu quả nếu mang vào ứng dụng thực tế. Để hiện thực hóa điều này nhóm cần cải tiến và phát triển thêm nhiều hơn nữa.

### Khó khăn khi xây dựng chương trình

Vì là lần đầu tiếp cận với ngôn ngữ C# và khái niệm mới là bảng băm, đây là những kiến thức cần thiết để thực hiện đồ án, nên nhóm đã mất nhiều thời gian để tìm hiểu ngôn ngữ mới và khái niệm bảng băm này. Bên cạnh là khó khăn khi tìm hiểu để viết một hàm băm tốt, tìm kiếm dữ liệu cho từ điển.

### Ưu khuyết điểm của từ điển

#### Ưu điểm

* Từ điển được thiết kế đơn giản, thân thiện với người dùng.
* Có tốc độ tìm kiếm nhanh nhờ vào bảng băm.
* Bảo toàn được dữ liệu trong quá trình nạp vì mỗi khi nạp xong một gói từ (từ và phần nghĩa) thì sẽ được lưu lại ngay.
* Ngoài dữ liệu từ đã có sẵn, có thể nạp thêm dữ liệu mới tùy vào người dùng, từ đó mỗi người dùng khác nhau sẽ có một thư viện dữ liệu từ riêng của mình.
* Có chức năng đọc từ giúp người dùng biết được cách phát âm của từ .

#### Khuyết điểm

* Nạp dữ liệu mới chưa được nhanh nếu số lượng dữ liệu nạp vào quá lớn, vì phải lưu lại ngay mỗi khi nạp được một phần tử.
* Phần phát âm còn phụ thuộc vào kết nối mạng.

### Hướng phát triển

Trong tương lại nhóm sẽ hoàn thiện phần nạp dữ liệu, từ đó giúp người dùng có thể dễ dàng nạp dữ liệu mới từ nguồn có sẵn tương thích với phần mềm, đồng thời nhóm sẽ hỗ trợ cập nhật phiên bản cho phần mềm và dữ liệu mới cho từ điển.

Đồng thời nhóm sẽ nghiên cứu thêm phần phát âm để từ điển có thể dùng chức năng phát âm khi ngoại tuyến, từ đó thuận tiện hơn cho người dùng. Tuy nhiên điều này không thực sự quá cần thiết vì với thời buổi hiện đại, đâu đâu cũng có kết nối mạng thì việc dùng mạng là điểu tất yếu. Nhưng để tránh việc API được dùng trong tương lai sẽ không còn được hỗ trợ nữa thì nhóm sẽ vẫn cố gắng hoàn thiện phần này.

# TÀI LIỆU THAM KHẢO

1. Peter, J.W., *PJW hash*, Wikipedia

Link: https://en.wikipedia.org/wiki/PJW\_hash\_function

[Đã truy cập: 27/09/2019]

1. Nguyễn Văn Hiếu, *Bảng băm – Hash tables*

Link: https://nguyenvanhieu.vn/bang-bam-hash-tables/

[Đã truy cập: 21/09/2019]

1. Arash Partow, *General Purpose Hash Function Algorithms*

Link: http://www.partow.net/programming/hashfunctions/index.html

[Đã truy cập: 19/09/2019]

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers Principles Techniques and Tools (2nd Edition)*, Addison Wesley, PEARSON, 2006