

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA ĐIỆN – ĐIỆN TỬ



ĐỒ ÁN TỐT NGHIỆP
Thiết kế và lập trình ứng dụng tra cứu từ điển
Anh – Việt, Việt – Anh.

Giảng viên hướng dẫn : **PGS. TS. NGUYỄN THANH HẢI**

Sinh viên thực hiện: **TRINH VĂN HIẾU**

Lớp : kĩ thuật điện tử và tin học công nghiệp

Khoá : k55

Hà Nội, tháng ... năm ...

SVTH: *Trịnh Văn Hiếu* Lớp: *Kỹ thuật điện tử và tin học công nghiệp*
GVHD: *PGS. TS. Nguyễn Thành Hải*

MỎ ĐẦU

1. Lý do chọn đề tài

Trong sự phát triển của khoa học thế kỷ 21, thời đại của công nghệ 4.0. công nghệ thông tin hiện nay là ngành phát triển nhanh nhất. Công nghệ thông tin của nước ta còn mới, song tốc độ phát triển của nó rất nhanh và mạnh, chiếm một vị trí quan trọng trong ngành khoa học công nghệ. Một trong những ứng dụng khá phổ biến ở nước ta hiện nay là lĩnh vực xây dựng các ứng dụng để phục vụ các bài toán thực tế, nhằm tăng năng suất, nhanh chóng và hiệu quả hơn. Xây dựng các ứng dụng này giúp cho những người dùng dễ tiếp cận, xử lý nhanh, và hiệu quả hơn. Đồng thời nó giúp cho những người phát triển phần mềm sau có thể nghiên cứu, phát triển, và hoàn thiện ứng dụng hơn. Vì vậy, tôi chọn đề tài “thiết kế và lập trình ứng dụng tra cứu từ điển Anh – Việt, Việt – Anh” làm đề tài nghiên cứu của tôi.

2. Mục tiêu đề tài

Xây dựng được ứng dụng tra cứu từ điển Anh – Việt, Việt – Anh bằng ngôn ngữ C++.

3. Phạm vi nghiên cứu

Phân tích yêu cầu, thiết kế, lập trình và kiểm tra ứng dụng.

4. Phương pháp nghiên cứu

Đọc tham khảo một số tài liệu về phương pháp lập trình C++, trên cơ sở đó tiến hành phân tích yêu cầu, thiết kế, lập trình về ứng dụng tra cứu từ điển Anh – Việt, Việt – Anh.

Tham khảo, quan sát các ứng dụng, phần mềm tra cứu từ điển Anh – Việt, Việt – Anh trên thực tế.

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ ỦNG DỤNG EVDICT	1
1.1 YÊU CẦU CỦA ỦNG DỤNG	1
1.1.1 Giới thiệu	1
1.1.2 Yêu cầu ứng dụng EVDict	1
1.2 HIỆN TRẠNG CÁC ỦNG DỤNG TRÊN THỰC TẾ	2
1.2.1 Lạc Việt	2
1.2.2 TFlat	3
1.2.3 LaBan Dict	4
1.2.4 Vlook	4
1.2.5 Dictionary.com	5
1.2.6 Oxford Dictionary	6
1.2.7 EVDict	6
1.3 NỘI DUNG CẦN LÀM	7
1.3.1 Cài đặt	7
1.3.2 Cách sử dụng	7
CHƯƠNG 2: MÔI TRƯỜNG LẬP TRÌNH ỦNG DỤNG	10
2.1 CƠ SỞ LÝ THUYẾT	10
2.1.1 Ngôn ngữ lập trình C++	10
2.1.1.1 Các khái niệm cơ bản	10
2.1.1.1.1 C++ là gì?	10
2.1.1.1.2 Cú pháp C++ cơ bản	11
2.1.1.1.3 Hằng, biến, kiểu dữ liệu	13
2.1.1.1.4 Toán tử trong C++	17
2.1.1.1.5 Vòng lặp	18
2.1.1.1.6 Mảng	19
2.1.1.1.7 Con trỏ	20
2.1.1.1.8 Hướng đối tượng trong C++	21
2.1.1.2 Lập trình Windows Destop Application	25
2.1.2.1 Tạo một project	25
2.1.2.2 Cấu trúc chương trình trong windows	26

CHƯƠNG 3 : THIẾT KẾ, LẬP TRÌNH ỨNG DỤNG	30
3.1 PHÂN TÍCH YÊU CẦU	30
3.1.1 Mô tả tổng quan	30
3.1.1.1 Các chức năng của chương trình	30
3.1.1.2 Sử dụng case diagram	30
3.1.1.3 Mô tả diagram	30
3.1.1.3.1 Use case 01 (UC01) – Search dictionary	30
3.1.1.3.2 Use case 02 (UC02) – Add word	31
3.1.1.3.3 Use case 03 (UC03): Delete word	32
3.1.1.3.4 Use case 04 (UC04): Update word	32
3.1.1.4 Mô tả luồng dữ liệu	33
3.1.1.4.1 Tìm kiếm từ	33
3.1.1.4.2 Thêm từ	34
3.1.1.4.3 Xóa từ	35
3.1.1.4.4 Cập nhật nghĩa của từ	36
3.1.2 Thiết kế	37
3.1.2.1 Giao diện hiển thị	37
3.1.2.2 Mô tả Menu bar	37
3.1.2.3 Mô tả Toolbar	38
3.1.2.4 Mô tả Combobox và Button	38
3.1.2.5 Mô tả Listbox	38
3.1.2.6 Mô tả Richedit	38
3.1.2.7 Mô tả Status bar	39
3.1.2.8 Mô tả Dialog box	39
3.2 LẬP TRÌNH	44
3.2.1 Class diagram	44
3.2.2 Lập trình hiển thị giao diện	47
3.2.3 Tiến hành lập trình	50
3.2.3.1 Lập trình hiển thị danh sách từ điển khi mở ứng dụng	50
3.2.3.2 Lập trình tính năng tra cứu từ điển	51
3.2.3.3 Lập trình tính năng xóa từ điển	54

3.2.3.4	Lập trình tính năng cập nhật nghĩa của từ.....	54
3.2.3.5	Lập trình tính năng thêm từ.....	55
3.2.3.6	Lập trình cài đặt các hot key (phím tắt)	55
3.3	KIỂM TRA, ĐÁNH GIÁ VÀ ĐÓNG GÓI ỨNG DỤNG	56
3.1.1	Kiểm tra, đánh giá ứng dụng.....	56
3.1.2	Đóng gói ứng dụng.....	56
KẾT LUẬN	57	
PHỤ LỤC	58	
DANH MỤC TÀI LIỆU THAM KHẢO	59	

DANH MỤC CÁC BẢNG BIỂU

BẢNG 2. 1: Kiểu biến, độ rộng và phạm vi	14
BẢNG 2. 2: Khởi tạo biến	16
BẢNG 2. 3: Toán tử số học	17
BẢNG 2. 4: Toán tử quan hệ	18
BẢNG 2. 5: Toán tử logic	18
BẢNG 2. 6: Vòng lặp.....	19
BẢNG 2. 7: Lệnh điều khiển vòng lặp	19
BẢNG 2. 8: Các message cơ bản.....	28
BẢNG 3. 1: Các case diagram	30
BẢNG 3. 2: Luồng thực thi tìm kiếm	33
BẢNG 3. 3: Luồng thực thi thêm từ	34
BẢNG 3. 4: Luồng thực thi xóa từ	35
BẢNG 3. 5: Luồng thực thi cập nhật từ	36
BẢNG 3. 6: Mô tả các class diagram và functions	46

DANH MỤC CÁC HÌNH VẼ

HÌNH 1. 1: Giao diện từ điển Lạc Việt	2
HÌNH 1. 2: Giao diện từ điển TFlat trên di động	3
HÌNH 1. 3: Giao diện từ điển Laban trên di động	4
HÌNH 1. 4: Giao diện từ điển Vlook	5
HÌNH 1. 5: Giao diện từ điển dictionary.com	6
HÌNH 1. 6: Giao diện từ điển oxford dictionary	6
HÌNH 1. 7: Giao diện lập trình ứng dụng EVDict	7
HÌNH 1. 8: Hộp thoại thông tin người lập trình ứng dụng	9
HÌNH 2. 1: Ví dụ cấu trúc của chương trình C++	13
HÌNH 2. 2: Comment trong C++	13
HÌNH 2. 3: Khai báo biến	15
HÌNH 2. 4: Khai báo biến toàn cục và cục bộ	16
HÌNH 2. 5: Con trỏ	21
HÌNH 2. 6: Tính đóng gói	23
HÌNH 2. 7: Đa hình tĩnh overloading	24
HÌNH 2. 8: Đa hình tĩnh overriding	24
HÌNH 2. 9: Đa hình động	25
HÌNH 2. 10: Đăng ký cửa sổ	27
HÌNH 2. 11: Tạo cửa sổ	27
HÌNH 2. 12: Vòng lặp message	28
HÌNH 2. 13: Sơ đồ xử lý message	29
HÌNH 3. 1: Use case 01: Search dictionary	31
HÌNH 3. 2: Use case 02: Add word	31
HÌNH 3. 3: Use case 03: Delete word	32
HÌNH 3. 4: Use case 04: Update word	32
HÌNH 3. 5: Sơ đồ luồng thực thi tìm kiếm	33
HÌNH 3. 6: Sơ đồ luồng thực thi thêm từ	34
HÌNH 3. 7: Sơ đồ luồng thực thi xóa từ	35
HÌNH 3. 8: Sơ đồ luồng thực thi cập nhật nghĩa của từ	36
HÌNH 3. 9: Giao diện chính của ứng dụng EVDict	37
HÌNH 3. 10: Hộp thoại thông tin	39
HÌNH 3. 11: Update Error	40
HÌNH 3. 12: Delete error	40
HÌNH 3. 13: Update success	41
HÌNH 3. 14: Delete success	41
HÌNH 3. 15: Add new word	42
HÌNH 3. 16: Add mean	42
HÌNH 3. 17: Lựa chọn chế độ xóa	43

HÌNH 3. 18: Delete word	43
HÌNH 3. 19: Destroy word	44
HÌNH 3. 20: Class diagram	44
HÌNH 3. 21: Tạo menu bar	47
HÌNH 3. 22: Hàm tạo cửa sổ	49
HÌNH 3. 23: Giao diện hiển thị sau khi tạo	50
HÌNH 3. 24: Giao diện hiển thị chính	51
HÌNH 3. 25: Xác định từ trong Listbox và mở file lấy nghĩa	52
HÌNH 3. 26: Tìm kiếm từ trên Listbox	53
HÌNH 3. 27: Tìm kiếm từ trên Combobox	53
HÌNH 3. 28: Lịch sử tìm kiếm trên Combobox	54
HÌNH 3. 29: Đăng ký các hot key	56

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

STT	Từ viết tắt	Mô tả
1.	UC	Use case
2.	User	Người sử dụng
3.	EVDict	Chương trình từ điển Anh – Việt, Việt – Anh.
4.	V – E	Việt – Anh
5.	E – V	Anh – Việt
6.	App	Ứng dụng
7.	Offline	Không cần kết nối mạng
8.	Online	Cần kết nối mạng
9.	OOP	Object-oriented programming (lập trình hướng đối tượng).
10.	Message	Tin nhắn
11.	File	Tệp
12.	Hot key	Tổ hợp phím

CHƯƠNG 1: TỔNG QUAN VỀ ỦNG DỤNG EVDICT

1.1 YÊU CẦU CỦA ỦNG DỤNG

1.1.1 Giới thiệu

Từ điển là danh sách các từ, ngữ được sắp xếp thành các từ vị chuẩn (*lemma*).

Một từ điển thông thường cung cấp các giải nghĩa các từ ngữ đó hoặc các từ ngữ tương đương trong một hay nhiều thứ tiếng khác.

Từ điển là nơi giải thích thông tin về ngôn ngữ của con người một cách dễ hiểu và khách quan nhất.

Ứng dụng là một chương trình mà người lập trình tạo ra, nhằm phục vụ một mục đích riêng nào đó. Nó thân thiện với người sử dụng, dễ hiểu, và rút ngắn thời gian thực hiện thủ công một việc nào đó.

EVDict là tên một ứng dụng được lập trình bằng ngôn ngữ lập trình. Nó giống như một quyển sách từ điển, và giúp người dùng thao tác, tra cứu nhanh gọn, dễ hiểu hơn.

1.1.2 Yêu cầu của ứng dụng *EVDict*

Giao diện (nơi giao tiếp, thao tác của người dùng lên ứng dụng) gồm các thành phần:

- Các chế độ, lựa chọn thao tác.
- Một phần hiển thị danh sách từ điển và một phần hiển thị nghĩa của từ tra được.
- Thông tin về ứng dụng.

Tính năng của ứng dụng:

- Thay đổi chế độ: người dùng có thể thay đổi chế độ tra cứu từ điển Anh – Việt, Việt – Anh bất kỳ khi nào muốn.
- Tra cứu từ điển: người dùng có thể tra cứu từ điển bằng cách chọn vào từ muốn tra cứu, hoặc là gõ từ cần tìm vào ô tìm kiếm.
- Phát âm các từ.

- Tra cứu từ chuyên ngành.

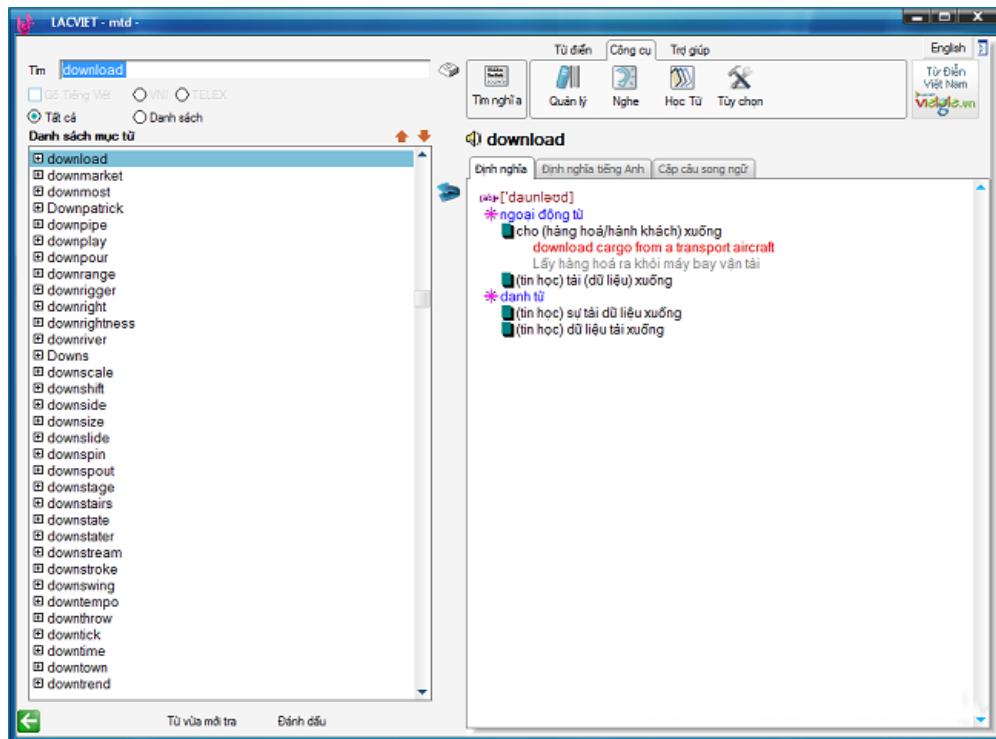
Dữ liệu: đa dạng, phong phú, từ và nghĩa phải đúng chuẩn.

1.2 HIỆN TRẠNG CÁC ỨNG DỤNG TRÊN THỰC TẾ

Hiện nay tiếng anh dường như là ngôn ngữ được nhiều người quan tâm nhất, đi theo đó không thể thiếu các công cụ tra cứu hay còn gọi là từ điển. Trước kia, công cụ tra cứu phổ biến nhất là *Lạc – Việt* thế nhưng khi thời đại công nghệ phát triển thì theo nó cũng sản sinh ra nhiều ứng dụng tra cứu từ điển phong phú hơn. Một số ứng dụng tra cứu từ điển nổi bật:

1.2.1 Lạc Việt

App từ điển đầu tiên phải kể đến là app *Lạc Việt*.



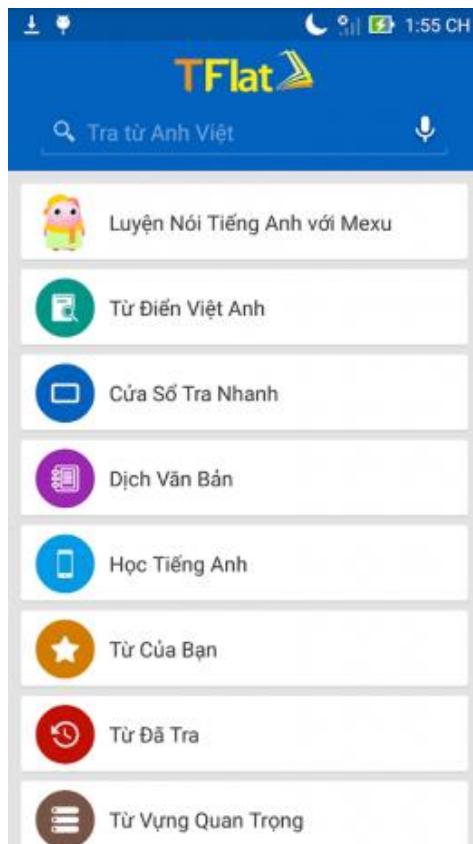
HÌNH 1. 1: Giao diện từ điển Lạc Việt

Chắc hẳn không còn ai xa lạc với phần mềm từ điển *Lạc Việt* đã từng “tung hoành” trên máy tính của mọi người. Lạc Việt đã hỗ trợ rất nhiều cho người dùng trên máy tính. Và nay, nó cũng phát triển thành app trên điện thoại. Lạc Việt có 3 bộ từ điển là: Anh – Việt, Việt – Anh, Việt – Việt với số lượng hơn 400,000 từ và cụm từ. Giao diện Lạc Việt dễ sử dụng và thao tác. Bên cạnh đó nó còn cho phép

người dùng tra cứu từ điển offline, phát âm từ vựng, thể hiện cụm từ liên quan, câu văn ví dụ, ... tạo sự thuận tiện khi tra từ. Ngoài ra *Lạc Việt* còn có phần lịch sử tự động sao lưu những từ đã tra. Người dùng có thể dịch đoạn văn từ tiếng anh sang tiếng việt khi kết nối mạng hỗ trợ đa tiện ích cho người sử dụng.

1.2.2 TFlat

TFlat là một trong số những ứng dụng được sử dụng nhiều nhất hiện nay.



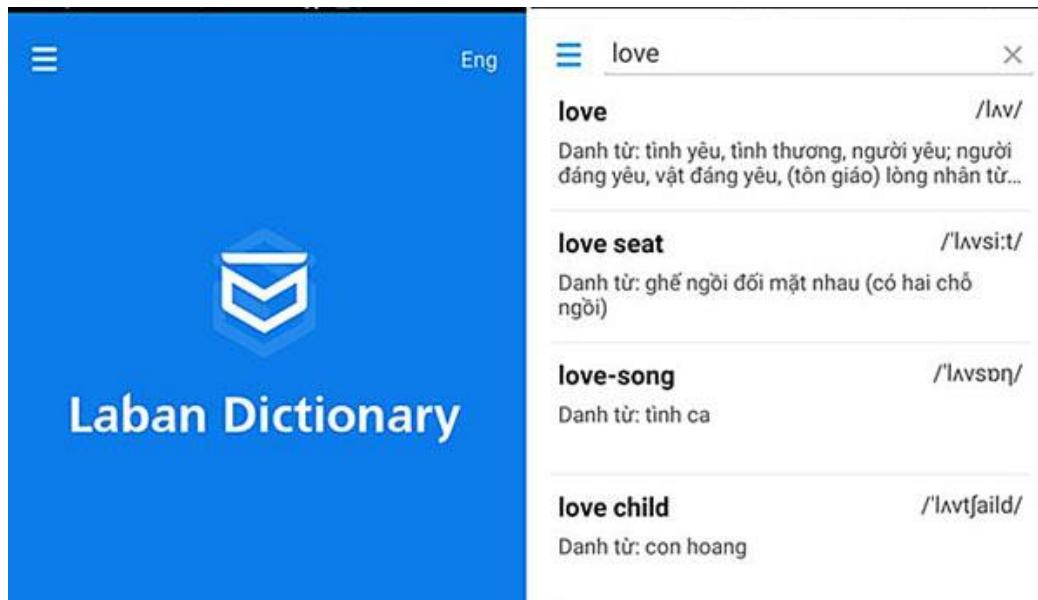
HÌNH 1. 2: Giao diện từ điển *TFlat* trên di động

TFlat có giao diện dễ sử dụng và thân thiện với người dùng. Nó cho phép người dùng tra cứu offline và online. Và cập nhật những từ mới qua kết nối mạng. Không chỉ vậy, *TFlat* còn có chức năng sao lưu từ đã tra, dịch câu văn, tra nhanh,...

Đi kèm với *TFlat* người dùng có thể tải các ứng dụng liên quan *Học tiếng Anh TFlat*, *Học phát âm tiếng Anh*, *Luyện nghe tiếng Anh TFlat*, *Ngữ pháp tiếng Anh*, ... và các ứng dụng này cũng hoàn toàn miễn phí và sử dụng offline.

1.2.3 LaBan Dict

Ứng dụng *LaBan* là lựa chọn tuyệt vời cho ứng dụng tra cứu từ điển trên thiết bị di động. *LaBan* không chỉ hỗ trợ người dùng tra cứu *offline* với trữ lượng 300,000 từ mà còn tạo ra sự thoải mái cho người dùng khi không có quảng cáo và hoàn toàn miễn phí



HÌNH 1. 3: Giao diện từ điển Laban trên di động

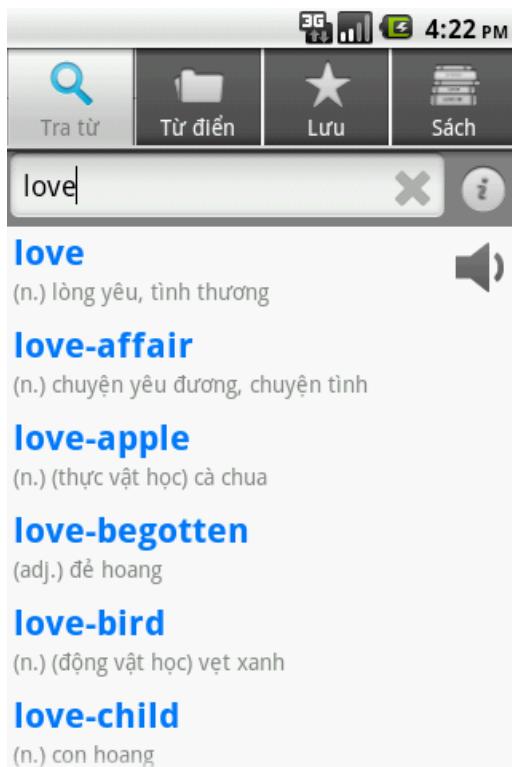
LaBan là app điện thoại do VNG Online phát triển nguồn dữ liệu từ cuốn từ điển Anh – Việt của giáo sư Lê Khả Kế và *LaBan* là app điện thoại có bản quyền từ Viện Ngôn Ngữ Học Việt Nam.

Bên cạnh những chức năng thường có trên app từ điển như tra từ nhanh, gợi ý từ gần giống, tự động sao lưu lịch sử tra từ, tra chéo giữa các bộ từ điển, ... *LaBan* còn sở hữu một số chức năng nổi bật như hỗ trợ tra từ bằng giọng nói và là ứng dụng đầu tiên có thể nhận dạng tra từ qua camera. *Laban* thực sự là ứng dụng tra từ điển không thể thiếu cho các thiết bị di động.

1.2.4 Vlook

Từ điển *Vlook* là bộ từ điển Anh – Việt được phát hành miễn phí trên Google Store và chỉ có thể tra từ Anh – Việt. *Vlook* có giao diện đơn giản hỗ trợ tra cứu offline không cần kết nối mạng với gần 60,000 từ trong đó có hơn 40,000 từ có

phiên âm và khoảng 3,000 từ thông dụng của từ điển Oxford. *Vlook* cho phép tìm kiếm từ, một phần của từ, mẫu câu và cho phép người dùng sao lưu từ vựng. Mặc dù có hỗ trợ phát âm từ vựng nhưng *Vlook* không hỗ trợ phát âm toàn bộ và chỉ một số từ vựng thông dụng.

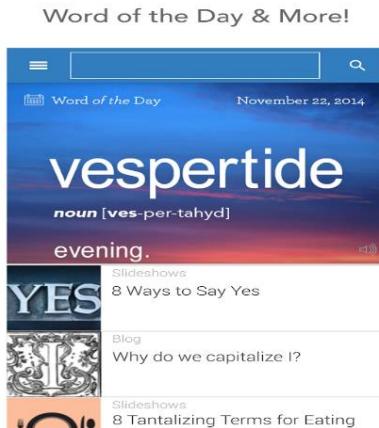


HÌNH 1. 4: Giao diện từ điển Vlook

1.2.5 Dictionary.com

Dictionary.com là ứng dụng từ điển Anh – Anh sở hữu trên 2,000,000 từ vựng. Ứng dụng cho phép người dùng sử dụng *online* và *offline* hoàn toàn miễn phí.

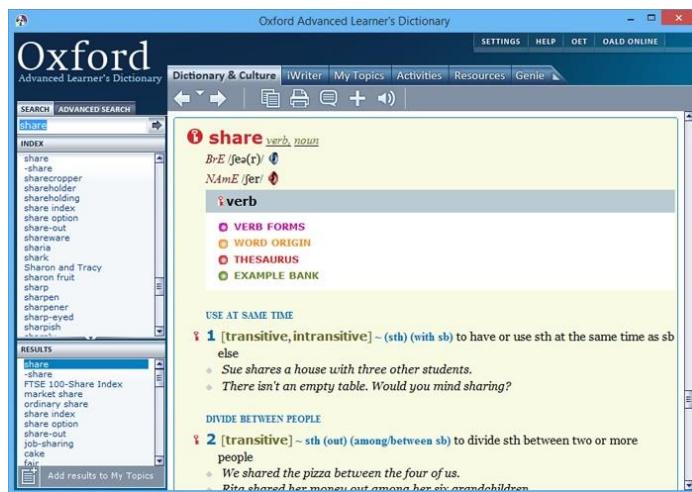
Bên cạnh những tính năng hữu ích như tra từ, phát âm từ vựng, tìm kiếm từ bằng giọng nói, gợi ý chính tả, tra thành ngữ và cụm từ,... ứng dụng có nhiều tiện ích giúp ích cho việc học từ như học từ mỗi ngày (*word of the day*) với hình ảnh, giải thích, phát âm và ví dụ sử dụng chi tiết, dễ hiểu.



HÌNH 1. 5: Giao diện từ điển dictionary.com

1.2.6 Oxford Dictionary

Đây là ứng dụng tra cứu từ điển chuyên nghiệp với khối lượng khổng lồ, trên 350,000 từ vựng. Ứng dụng *Oxford Dictionary of English* là phần mềm tra từ Anh – Anh bao gồm mọi từ vựng của nhiều chuyên ngành, cách sử dụng từ,... Ứng dụng còn có chức năng tra từ qua camera, các từ mỗi ngày,... giúp người dùng trau dồi thêm từ vựng tiếng anh. Tuy nhiên ứng dụng còn có các banner quảng cáo. Người dùng có thể sử dụng bản *online* của ứng dụng từ điển hoàn toàn miễn phí và trả mức phí để sử dụng bản *offline*.



HÌNH 1. 6: Giao diện từ điển oxford dictionary

1.2.7 EVDict

Và cuối cùng không thể không kể đến là ứng dụng *EVDict*. *EVDict* không chỉ giúp người dùng tra cứu *offline* với số lượng từ “khủng” 380,000 từ mà còn hỗ trợ

phát âm với hơn 10,000 từ thông dụng. Ngoài ra, ứng dụng còn có số lượng từ chuyên ngành và cho phép người dùng tìm kiếm theo từ, và theo một phần của từ, mẫu câu...

1.3 NỘI DUNG CẦN LÀM

Tài liệu này sẽ thiết kế và lập trình ứng dụng tra cứu từ điển Anh – Việt, Việt – Anh. Ứng dụng này sẽ bao gồm:

- Giao diện hiển thị.
 - Các tính năng:
 - Tra cứu từ: chọn từ hoặc gõ từ tìm kiếm.
 - Thêm từ: thêm từ và nghĩa mới vào từ điển.
 - Sửa: thay đổi nghĩa của từ.
 - Xóa: xóa từ.

1.3.1 Cài đặt

Cài đặt như một ứng dụng windows bình thường.

1.3.2 Cách sử dụng

Sau khi mở, ứng dụng sẽ có giao diện như *Hình 1.7*:



HÌNH 1. 7: Giao diện lập trình ứng dụng EVDict

Chọn chế độ: ban đầu khi mở ứng dụng lên, sẽ mặc định ở chế độ tra cứu từ điển Anh – Việt. Người sử dụng có thể thay đổi bằng cách:

- Ché độ Anh – Việt: từ thanh menu, chọn Dictionary → Eng – Viet (hoặc phím tắt: Alt + D + E). Hoặc cũng có thể sử dụng thanh công cụ bằng cách: chọn vào biểu tượng chữ E.
 - Ché độ Việt – Anh: từ thanh menu, chọn Dictionary → Viet – Eng (hoặc phím tắt: Alt + D + V). Hoặc cũng có thể sử dụng thanh công cụ: chọn biểu tượng chữ V.

Tra cứu từ: người dùng có thể tra cứu từ bằng các cách:

- Chọn trực tiếp trên danh sách từ điển (double click), sau khi chọn, nghĩa của từ đó sẽ được hiện trên giao diện hiển thị.
 - Người dùng có thể gõ từ muốn tra cứu vào ô tra cứu, sau khi gõ xong, bấm vào nút tra cứu, có biểu tượng “>>”. Nếu từ gõ đúng thì nghĩa của từ đó sẽ hiển thị trên giao diện hiển thị. Nếu từ đó sai, hoặc không nằm trong danh sách từ điển của ứng dụng, sẽ có thông báo cho người dùng.

Chỉnh sửa nghĩa của từ: người dùng có thể thay đổi nghĩa của từ sau khi tìm kiếm, thay đổi trực tiếp trên giao diện hiển thị nghĩa, sau đó cập nhật nghĩa của từ đó vào danh sách từ điển của ứng dụng bằng cách:

- Từ thanh menu: chọn Menu → Update (hoặc phím tắt: Alt + M + U).
 - Từ thanh công cụ: chọn trực tiếp vào biểu tượng chiếc bút để update.

Xóa từ: người dùng có thể chọn từ rồi xóa từ đó bằng cách:

- Từ thanh menu: chọn Menu → Delete (hoặc phím tắt: Alt + M + D).
 - Từ thanh công cụ: bấm vào biểu tượng chữ “X”.

Sau khi chọn xóa từ, sẽ có hộp thoại hiển thị thông báo và có ba lựa chọn cho người dùng:

- Destroy: sẽ xóa từ hoàn toàn trong ứng dụng, lần sau mở lại từ đó cũng còn nữa.
 - Delete: xóa tạm từ trên ứng dụng, khi mở lại ứng dụng, từ đó vẫn còn.
 - Cancel: không làm gì

Thêm từ: người sử dụng có thể thêm bằng cách:

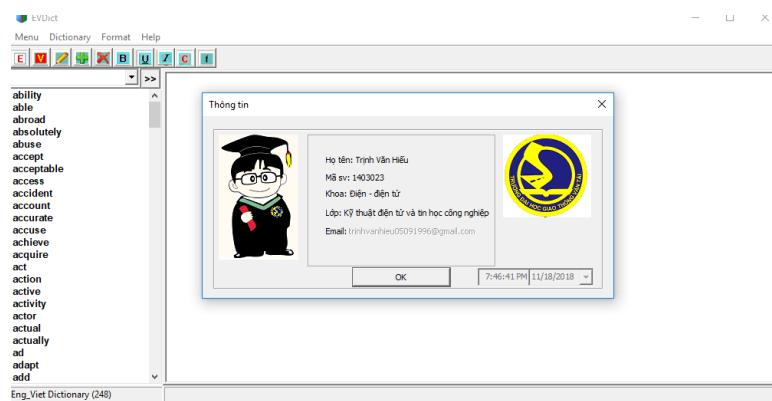
- Từ thanh menu: chọn Menu → Add (hoặc phím tắt: Alt + M + A).
- Từ thanh công cụ: chọn vào biểu tượng dấu cộng.

Sau khi chọn thêm từ, sẽ có một hộp thoại hiện ra, để người dùng nhập từ cần thêm vào. Nếu từ đó:

- Đã có trong danh sách từ điển, thì sẽ có một hộp thoại lựa chọn và thông báo cho người dùng.
- Đã bị xóa bằng chế độ xóa tạm: thì từ đó sẽ được khôi phục lại, và có thông báo cho người dùng.
- Nếu từ đó chưa có trong danh sách, thì sẽ có thông báo, và yêu cầu người dùng nhập nghĩa của từ đó vào trong giao diện hiện thị rồi chọn Update để hoàn thiện thêm từ.

Xem thông tin người lập trình: khi mở ứng dụng lên, sẽ có một hộp thoại hiển thị thông tin người lập trình, hoặc cũng có thể chọn từ thanh menu: chọn Help → About (hoặc phím tắt: Alt + H + A). Thông tin cụ thể của ứng dụng này bao gồm:

- Họ tên: Trịnh Văn Hiếu
- Mã sv: 1403023
- Khoa : Điện – điện tử
- Lớp: Kỹ thuật điện tử và tin học công nghiệp
- Email: trinhvanhieutamthuan05091996@gmail.com
- Biểu tượng: trường đại học Giao Thông Vận Tải hà nội
- Thông tin về: ngày – tháng – năm, giờ tra cứu thông tin.



HÌNH 1. 8: hộp thoại thông tin người lập trình ứng dụng

CHƯƠNG 2: MÔI TRƯỜNG LẬP TRÌNH ỨNG DỤNG

2.1 CƠ SỞ LÝ THUYẾT

Trong giai đoạn hiện nay, giai đoạn mà nền công nghiệp công nghệ thông tin chiếm một vị trí hết sức quan trọng. Mọi quốc gia trên thế giới đều cố gắng nỗ lực vào, và phấn đấu phát triển mạnh lĩnh vực công nghệ thông tin. Một trong số những công cụ tiếp cận nhanh nhất đó là tin học.

Đã qua rồi, thời mà ông cha ta phải làm và ghi nhớ mọi việc bằng những phương tiện thô sơ. Để tính toán, họ đều bắt đầu bằng tay và tính toán một cách chi tiết. Còn ngay nay, thời đại của công nghệ thông tin. Bạn không cần phải làm như vậy nữa, bạn chỉ cần lập ra một chương trình, một ứng dụng rồi sử dụng nó. Nó tiện ích và thuận tiện hơn rất nhiều, nó không bị hạn chế về khối lượng, con số. Thậm chí, trong vài giây nó có thể giải quyết hàng vạn bài toán phức tạp. Do đó, tốc độ làm việc và năng suất tăng lên rất nhiều. Để nó có thể làm việc và hoạt động được thì chúng ta phải lập trình, mà muốn lập trình thì chúng ta cần có ngôn ngữ lập trình. Có rất nhiều ngôn ngữ lập trình như *Pascal*, *Cobol*, *C++*.... Mỗi ngôn ngữ đều có chức năng riêng, thế mạnh riêng. Trong đó, ngôn ngữ *C++* có thể mạnh về can thiệp sâu vào máy tính, lại dễ nhìn dễ học. Chính vì vậy, tôi chọn ngôn ngữ này để lập trình ứng dụng tra cứu từ điển trong đồ án của tôi. Ngôn ngữ lập trình thể hiện tư duy, thuật toán, cách hoạt động của chương trình. Và đặc biệt, chương trình đó phải chạy trên một môi trường như thế nào? Dựa vào yêu cầu của ứng dụng tra cứu từ điển, tôi chọn lập trình *C++* trên môi trường *Windows Destop Application*. Môi trường này rất hữu ích, nó sử dụng ngôn ngữ *C++* để lập trình. Bên cạnh đó, nó hỗ trợ rất nhiều trong quá trình gợi ý, tìm và sửa lỗi, và thiết kế giao diện hiển thị.

2.1.1 Ngôn ngữ lập trình C++

2.1.1.1 Các khái niệm cơ bản

2.1.1.1.1 C++ là gì?

C++ là một ngôn ngữ lập trình kiêu tĩnh, dữ liệu trừ tượng, phân biệt kiểu chữ thường chữ hoa mà hỗ trợ lập trình hướng đối tượng, lập trình thủ tục.

C++ được coi như là ngôn ngữ bậc trung (*middle-level*), khi nó kết hợp các đặc điểm và tính năng của ngôn ngữ bậc cao và bậc thấp.

C++ được phát triển bởi Bjarne Stroustrup năm 1979 tại Bell Labs ở Murray Hill, New Jersey, như là một bản nâng cao của ngôn ngữ C và với tên gọi đầu tiên là "C với các Lớp", nhưng sau đó được đổi tên thành C++ vào năm 1983.

C++ là một Superset của C, và bất kỳ chương trình C nào cũng là một chương trình C++.

C++ hỗ trợ đầy đủ lập trình hướng đối tượng, bao gồm 4 tính năng trụ cột của lập trình hướng đối tượng là:

- Tính trừu tượng
- Tính đóng gói
- Tính kế thừa
- Tính đa hình

C++ chuẩn gồm 3 phần quan trọng:

- Core Language cung cấp tất cả các khái bao gồm biến, kiểu dữ liệu (data type) và literals, ...
- Thư viện chuẩn C++ (C++ Standard Library) cung cấp tập hợp hàm đa dạng để thao tác file, string, ...
- Standard Template Library (STL) cung cấp tập hợp phương thức đa dạng để thao tác cấu trúc dữ liệu, ...

2.1.1.1.2 Cú pháp C++ cơ bản

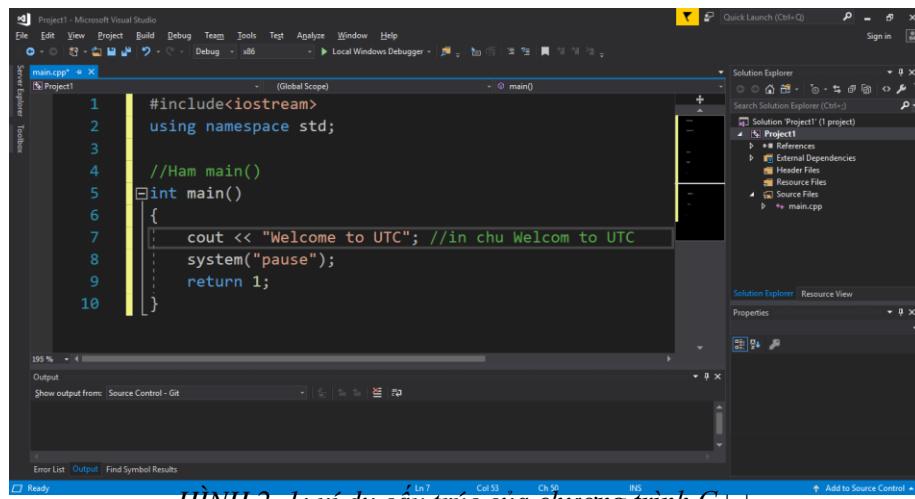
Khi chúng ta xem xét một chương trình C++, nó có thể được định nghĩa như là một tập hợp của các đối tượng, mà giao tiếp thông qua việc triệu hồi các phương thức của mỗi đối tượng đó. Dưới đây, chúng tôi miêu tả ngắn gọn ý nghĩa của *class* (lớp), *object* (đối tượng), *method* (phương thức) và các biến đối tượng:

- Đối tượng: Đối tượng có các trạng thái và hành vi. Ví dụ: một đối tượng dog có các trạng thái là *color*, *name*, *breed*, và các hành vi là *wagging*, *barking*, *eating*. Một đối tượng là một minh họa của một lớp.

- Lớp: Một lớp có thể được định nghĩa như là một *template/blueprint*, mà miêu tả hành vi/trạng thái mà đối tượng hỗ trợ.
- Phương thức: Về cơ bản, một phương thức là một hành vi. Một lớp có thể chứa nhiều phương thức. Phương thức là nơi tính *logic* được viết, dữ liệu được thao tác và tất cả *action* được thực thi.
- Biến *instance*: Mỗi đối tượng có tập hợp biến đối tượng duy nhất của nó. Trạng thái của một đối tượng được tạo ra bởi các giá trị được gán cho các biến đối tượng của nó.

Cấu trúc của một chương trình gồm cách phân *Hình 2.1*:

- Ngôn ngữ C++ định nghĩa một số header, mà chứa thông tin cần thiết và hữu ích cho chương trình của bạn. Với chương trình này, header là *<iostream>* là cần thiết.
- Dòng *using namespace std;* nói cho compiler sử dụng std namespace. Namespace là phần bổ sung gần đây cho C++.
- Dòng tiếp theo là một chú thích đơn dòng trong C++. Các chú thích đơn dòng bắt đầu với // và kết thúc ở cuối dòng.
- Dòng *int main()* là hàm *main*, tại đây việc thực thi chương trình bắt đầu.
- Dòng tiếp theo *cout << "Welcome to UTC";* để in dòng chữ "Welcome to UTC" trên màn hình.
- Dòng tiếp theo *return 0;* kết thúc hàm *main()* và làm nó trả về giá trị 0 tới tiến trình đang gọi.

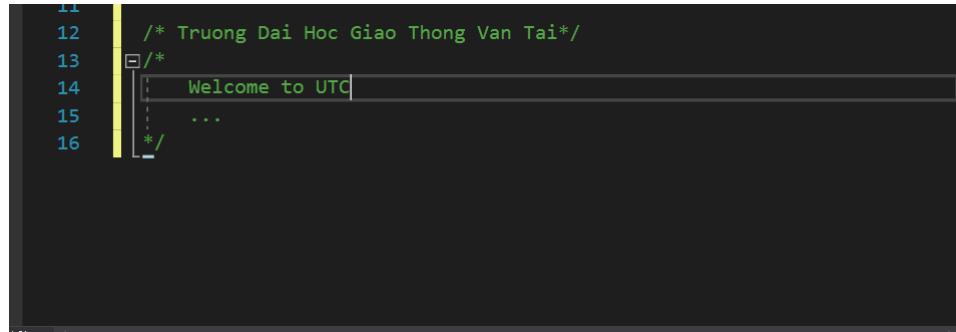


HÌNH 2. 1: ví dụ cấu trúc của chương trình C++

Comment của chương trình là các lời diễn giải, mà bạn có thể bao trong C/C++ code, và giúp cho bất kỳ ai đọc source code dễ dàng hơn. Tất cả ngôn ngữ lập trình đều cho phép một số mẫu *comment* nào đó.

C++ hỗ trợ các *comment* đơn dòng và đa dòng. Tất cả ký tự có trong *comment* được bỏ qua bởi C/C++ *compiler*.

Comment trong C/C++ bắt đầu với `/*` và kết thúc với `*/`. Ví dụ:



HÌNH 2. 2: Comment trong C++

2.1.1.3 Hằng, biến, kiểu dữ liệu

a. Kiểu dữ liệu trong C/C++

Kiểu dữ liệu trong C/C++. Trong khi làm việc với bất kỳ ngôn ngữ lập trình nào, bạn cần sử dụng các kiểu biến đa dạng để lưu giữ thông tin. Các biến, không gì khác ngoài các vị trí bộ nhớ được dành riêng để lưu giá trị. Nghĩa là, khi bạn tạo một biến, bạn dành riêng một số không gian trong bộ nhớ cho biến đó.

Bạn có thể thích lưu thông tin của các kiểu dữ liệu (*Data Type*) đa dạng như *Character*, *Wide Character*, *integer*, *floating-point*, *double floating point*, *Boolean*, ... Dựa trên kiểu dữ liệu của một biến, hệ thống sẽ cấp phát bộ nhớ và quyết định những gì có thể được lưu giữ trong bộ nhớ dành riêng đó.

Kiểu dữ liệu nguyên thủy:

Kiểu	Độ rộng bit	Dãy giá trị
Char	1 byte	-127 tới 127 hoặc 0 tới 255
unsigned char	1 byte	0 tới 255
signed char	1 byte	-127 tới 127
Int	4 byte	-2147483648 tới 2147483647
unsigned int	4 byte	0 tới 4294967295
signed int	4 byte	-2147483648 tới 2147483647
short int	2 byte	-32768 tới 32767
unsigned short int	Range	0 tới 65,535
signed short int	Range	-32768 tới 32767
long int	4 byte	-2,147,483,647 tới 2,147,483,647
signed long int	4 byte	Tương tự như long int
unsigned long int	4 byte	0 tới 4,294,967,295
Float	4 byte	+/- 3.4e +/- 38 (~7 chữ số)
Double	8 byte	+/- 1.7e +/- 308 (~15 chữ số)
long double	8 byte	+/- 1.7e +/- 308 (~15 chữ số)
wchar_t	2 hoặc 4 byte	1 wide character

BẢNG 2. 1: kiểu biến, độ rộng và phạm vi

b. Kiểu biến trong C/C++

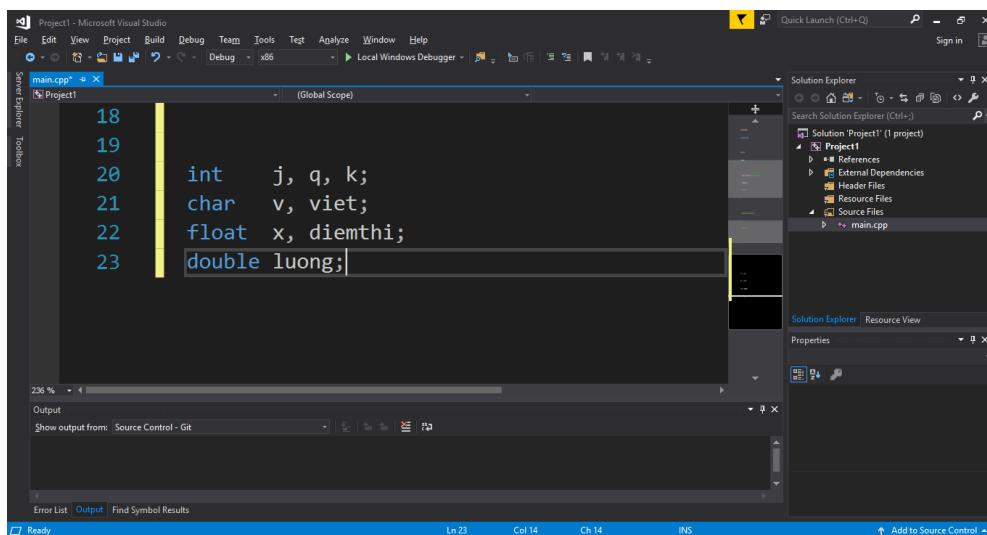
Một biến cung cấp nơi lưu giữ được đặt tên để chúng ta có thể thao tác. Mỗi biến trong C/C++ có một kiểu cụ thể, mà quyết định: kích cỡ và cách bố trí bộ nhớ của biến; dãy giá trị có thể được lưu giữ bên trong bộ nhớ đó; và tập hợp hoạt động có thể được áp dụng cho biến đó.

Tên biến có thể gồm các ký tự, các chữ số, dấu gạch dưới. Nó phải bắt đầu bởi hoặc một ký tự hoặc một dấu gạch dưới. Các ký tự chữ hoa và chữ thường là khác nhau bởi C/C++ là ngôn ngữ phân biệt kiểu chữ. Biến có thể trong các kiểu giá trị đa dạng như kiểu *char*, *int*, *float*, ...

Định nghĩa biến trong C/C++ nghĩa là nói cho compiler nơi và lượng bộ nhớ cần tạo để lưu giữ biến đó. Một định nghĩa biến xác định một kiểu dữ liệu, và chứa danh sách của một hoặc nhiều biến có kiểu đó, như sau:

kieu_gia_tri danh_sach_bien ;

Ở đây, *kieu_gia_tri* phải là một kiểu dữ liệu hợp lệ trong C/C++, gồm *char*, *w_char*, *int*, *float*, *double*, *bool* hoặc bất kỳ đối tượng nào mà người dùng tự định nghĩa, ... và *danh_sach_bien* có thể chứa một hoặc nhiều tên Identifier (Định danh) phân biệt nhau bởi dấu phẩy. Sau đây là một số khai báo hợp lệ trong C/C++:



HÌNH 2. 3: khai báo biến

c. Phạm vi biến

Một *scope* (phạm vi) là một khu vực của chương trình nơi biến hoạt động, và nói chung có thể có 3 khu vực mà biến có thể được khai báo:

Bên trong một hàm hoặc một khối, được gọi là biến cục bộ (*local*).

Trong định nghĩa của các tham số hàm, được gọi là các tham số chính thức (*formal*).

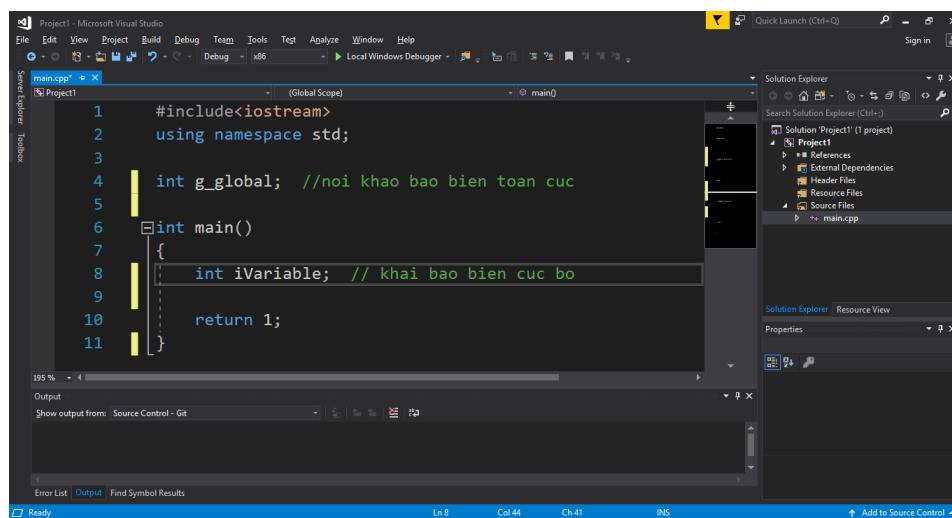
Bên ngoài của tất cả hàm, được gọi là biến toàn cục (*global*).

Các biến được khai báo bên trong một hàm hoặc khối là các biến cục bộ (*local*). Chúng chỉ có thể được sử dụng bởi các lệnh bên trong hàm hoặc khối *code*

đó. Các biến cục bộ không được biết ở bên ngoài hàm đó (tức là chỉ được sử dụng bên trong hàm hoặc khỏi *code* đó).

Biến toàn cục (*global*) trong C++ được định nghĩa bên ngoài các hàm, thường ở phần đầu chương trình. Các biến toàn cục giữ giá trị của nó trong suốt vòng đời chương trình của bạn.

Một biến toàn cục có thể được truy cập bởi bất kỳ hàm nào. Tức là, một biến toàn cục là có sẵn cho bạn sử dụng trong toàn bộ chương trình sau khi đã khai báo nó.



HÌNH 2. 4: khai báo biến toàn cục và cục bộ

Khi một biến cục bộ được định nghĩa, nó không được khởi tạo bởi hệ thống, chính bạn phải khởi tạo nó. Các biến toàn cục được khởi tạo tự động bởi hệ thống khi bạn định nghĩa chúng, như *Bảng 2.2*.

Khởi tạo biến một cách chính xác là một sự thực hành tốt, nếu không, đôi khi chương trình sẽ cho kết quả không mong đợi.

Kiểu dữ liệu	Giá trị khởi tạo
int	0
char	'\0'
float	0
double	0
pointer	NULL

BẢNG 2. 2: khởi tạo biến

d. Hằng (*constant/literal*) trong C/C++

Constant liên quan tới các giá trị cố định mà chương trình không thể thay đổi và chúng được gọi là *literals*.

Constant là một kiểu dữ liệu thay thế cho *Literal*, còn *Literal* thay hiện chính nó.

Trong ví dụ: `const PI = 3.14` thì *Constant* ở đây là *PI*, còn *Literal* là 3.14.

Constant có thể là bất kỳ kiểu dữ liệu cơ bản nào trong C/C++, và có thể được phân chia thành giá trị hằng số nguyên, hằng số thực, hằng ký tự, hằng chuỗi và *Boolean literal* (tạm dịch: hằng logic).

Ngoài ra, constant được đổi xử gióng như các biến thông thường, ngoại trừ việc giá trị của chúng là không thể thay đổi sau khi định nghĩa.

2.1.1.1.4 Toán tử trong C++

Một toán tử là một biểu tượng, mà nói cho compiler thực hiện các thao tác toán học và logic cụ thể. C++ cung cấp nhiều toán tử có sẵn, đó là:

- Toán tử số học, toán tử logic
- Toán tử quan hệ
- Toán tử so sánh bit
- Toán tử gán
- Toán tử hỗn hợp

a. Toán tử số học trong C++

Bảng 2.3 liệt kê các toán tử số học được hỗ trợ bởi ngôn ngữ C++.

Giả sử biến A giữ giá trị 10, biến B giữ 20 thì:

Toán tử	Miêu tả	Ví dụ
+	Cộng hai toán hạng	$A + B$ kết quả là 30
-	Trừ toán hạng thứ hai từ toán hạng đầu	$A - B$ kết quả là -10
*	Nhân hai toán hạng	$A * B$ kết quả là 200
/	Phép chia	B / A kết quả là 2
%	Phép lấy số dư	$B \% A$ kết quả là 0
++	Toán tử tăng (++), tăng giá trị toán hạng thêm một đơn vị	$A++$ kết quả là 11
--	Toán tử giảm (--), giảm giá trị toán hạng đi một đơn vị	$A--$ kết quả là 9

BẢNG 2. 3: toán tử số học

b. Toán tử quan hệ trong C++

Bảng 2.5 liệt kê các toán tử quan hệ được hỗ trợ bởi ngôn ngữ C++.

Giả sử biến A giữ giá trị 10, biến B giữ 20 thì:

Toán tử	Miêu tả	Ví dụ
<code>==</code>	nếu bằng nhau thì điều kiện là true	$(A == B)$ là không đúng
<code>!=</code>	Nếu không bằng thì điều kiện là true.	$(A != B)$ là true
<code>></code>	Nếu lớn hơn thì điều kiện là true.	$(A > B)$ là không đúng
<code><</code>	Nếu nhỏ hơn thì là true.	$(A < B)$ là true
<code>>=</code>	Nếu đúng là true.	$(A >= B)$ là không đúng
<code><=</code>	Nếu đúng là true.	$(A <= B)$ là true

BẢNG 2. 4: toán tử quan hệ

c. Toán tử logic

Toán tử	Miêu tả	Ví dụ
<code>&&</code>	Được gọi là toán tử logic AND (và). Nếu cả hai toán tử đều có giá trị khác 0 thì điều kiện trở lên true.	$(A \&\& B)$ là false.
<code> </code>	Được gọi là toán tử logic OR (hoặc). Nếu một trong hai toán tử khác 0, thì điều kiện là true.	$(A B)$ là true.
<code>!</code>	Được gọi là toán tử NOT (phủ định). Sử dụng để đảo ngược lại trạng thái logic của toán hạng đó. Nếu điều kiện toán hạng là true thì phủ định nó sẽ là false.	$!(A \&\& B)$ là true.

BẢNG 2.5: toán tử logic

2.1.1.1.5 Vòng lặp

Có một tình huống mà bạn cần phải thực hiện một đoạn code một vài lần. Nhìn chung, các câu lệnh được thực hiện một cách tuần tự. Câu lệnh đầu tiên của hàm được thực hiện trước, sau đó đến câu thứ 2 và tiếp tục

Ngôn ngữ lập trình cung cấp cho chúng ta nhiều cấu trúc điều khiển và cho phép bạn thực hiện những phần phức tạp.

Vòng lặp	Miêu tả
While	Lặp lại một hoặc một nhóm các lệnh trong khi điều kiện đã cho là đúng. Nó kiểm tra điều kiện trước khi thực hiện thân vòng lặp.
For	Thực thi một dãy các lệnh nhiều lần và tóm tắt đoạn code mà quản lý biến vòng lặp.
Do ... while	Giống lệnh While, ngoại trừ ở điểm là nó kiểm tra điều kiện ở cuối thân vòng lặp.
Lồng vòng lặp	Bạn có thể sử dụng một hoặc nhiều vòng lặp trong các vòng lặp while, for hoặc do..while khác.

BẢNG 2. 6: vòng lặp

Lệnh điều khiển	Miêu tả
Break	Kết thúc vòng lặp hoặc lệnh switch và chuyển sang thực thi vòng lặp hoặc lệnh switch ngay sau nó.
Continue	Khi gặp lệnh này thì chương trình sẽ bỏ qua các câu lệnh ở dưới nó (trong cùng một câu lệnh lặp) để thực hiện vòng lặp mới.
Goto	Chuyển tới lệnh được gán. Mặc dù vậy, nó được khuyên rằng không nên sử dụng lệnh goto trong chương trình của bạn.

BẢNG 2. 7: lệnh điều khiển vòng lặp

2.1.1.6 Mảng

Ngôn ngữ lập trình C/C++ cung cấp cấu trúc dữ liệu gọi là *mảng*, được lưu trữ trong một tập hợp các dữ liệu cùng kiểu với độ dài cố định. Một *mảng* được sử dụng để lưu trữ tập hợp dữ liệu, nhưng nó rất hữu dụng nếu bạn nghĩ về một *mảng* các biến với cùng một kiểu.

Thay vì khai báo biến một cách rời rạc, như biến so0, so1,... và so99, bạn có thể khai báo một *mảng* các giá trị như so[0], so[1] và ... so[99] để biểu diễn các

giá trị riêng biệt. Một thành viên cụ thể của *mảng* có thể được truy cập qua *index* (chỉ số).

Tất cả mảng đều bao gồm các vị trí nhô lièn kề nhau. Địa chỉ thấp nhất tương ứng với thành viên đầu tiên và địa chỉ cao nhất tương ứng với thành viên cuối cùng của *mảng*.

Để khai báo một mảng trong ngôn ngữ C/C++, bạn xác định kiểu của biến và số lượng các phần tử được yêu cầu bởi biến đó như sau:

Kieu Ten_mang [kich_co_mang];

Đây là *mảng* một chiều. *Kich_co_mang* phải là một số nguyên lớn hơn 0 và *Kieu* phải hợp lệ trong ngôn ngữ C/C++.

Bạn có thể khởi tạo mảng trong C/C++ hoặc từng phần tử một hoặc sử dụng một câu lệnh như dưới đây:

```
int a[3] = {1, 2, 3};
```

Số lượng các giá trị trong dấu ngoặc kép {} không được lớn hơn số lượng phần tử khai báo trong dấu ngoặc vuông [].

Nếu bạn bỏ sót kích cỡ mảng thì mảng đó đủ lớn để giữ các giá trị được khởi tạo: Bạn sẽ tạo chính xác một chuỗi có giá trị giống hệt chuỗi bên trên bằng cách gán từng phần tử một. Dưới đây là một ví dụ khi gán giá trị cho một phần tử của mảng:

```
int a[] = {1, 2, 3};
```

Một mảng được truy cập bởi cách đánh chỉ số trong tên của mảng.

2.1.1.1.7 Con trỏ

Một con trỏ là một biến mà trong đó giá trị của nó là địa chỉ của biến khác. Ví dụ như địa chỉ của vùng nhớ. Giống như các biến và hằng số, bạn phải khai báo con trỏ trước khi bạn có thể sử dụng nó để lưu trữ bất kì địa chỉ của biến nào. Dạng tổng quát của việc khai báo con trỏ như sau:

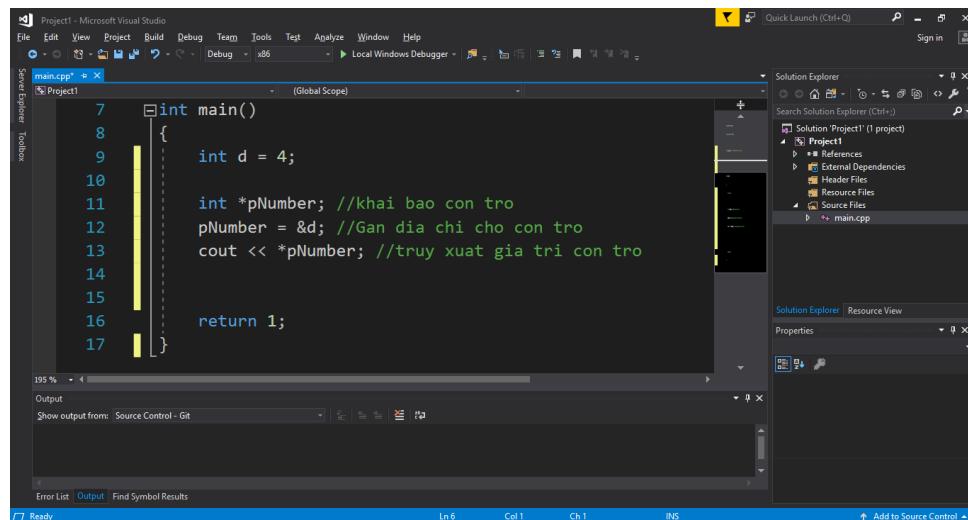
*kieu du lieu *ten bien;*

Ở đây, *kieu_du_lieu* là kiểu dữ liệu cơ bản con trỏ, nó là kiểu hợp lệ trong ngôn ngữ C và *ten_bien* là tên giá trị của con trỏ. Phần ký tự * sử dụng trong khai báo con trỏ giống như việc bạn sử dụng cho phép nhân. Mặc dù vậy, trong khai báo này, ký tự * được thiết kế để sử dụng các biến của con trỏ.

Cách sử dụng con trỏ:

- Khởi tạo và khai báo một con trỏ
- Gán địa chỉ
- Truy xuất giá trị

Điều này được thực hiện bởi toán tử * trả về giá trị các biến chứa trong địa chỉ được xác định bởi toán tử này. Dưới đây là các sử dụng những phép toán trên:



```

7 int main()
8 {
9     int d = 4;
10
11     int *pNumber; //khai bao con tro
12     pNumber = &d; //Gan dia chi cho con tro
13     cout << *pNumber; //truy xuat gia tri con tro
14
15
16     return 1;
17 }
```

HÌNH 2. 5: con trỏ

2.1.1.1.8 Hướng đối tượng trong C++

a. Lớp và đối tượng

Một định nghĩa lớp trong C++ bắt đầu với từ khóa *class*, được sau bởi tên lớp và phần thân lớp, được bao quanh trong một cặp dấu ngoặc mỏc. Một định nghĩa lớp phải được theo sau: hoặc bởi một dấu chấm phẩy hoặc một danh sách các khai báo.

Từ khóa *public* quyết định các thuộc tính truy cập của các thành viên lớp mà theo sau nó. Một thành viên *public* có thể được truy cập từ bên ngoài lớp bắt đầu bên trong phạm vi (*scope*) của đối tượng lớp đó.

Các thuộc tính có từ khóa *protected* chỉ có thể được truy xuất bởi lớp kế thừa từ lớp cơ sở.

Các thuộc tính có từ khóa *private* không thể truy xuất bởi bất kỳ đâu, muốn truy suất thuộc tính *private*, ta chỉ có thể truy xuất thông qua các hàm: *get()*, *set()*.

Các tính chất của lập trình hướng đối tượng:

- Tính trừu tượng.
- Tính đóng gói.
- Tính kế thừa.
- Tính đa hình.

b. Tính trừu tượng

Trừu tượng hóa dữ liệu (*Data abstraction*) liên quan tới việc chỉ cung cấp thông tin cần thiết tới bên ngoài và ẩn chi tiết cơ sở của chúng.

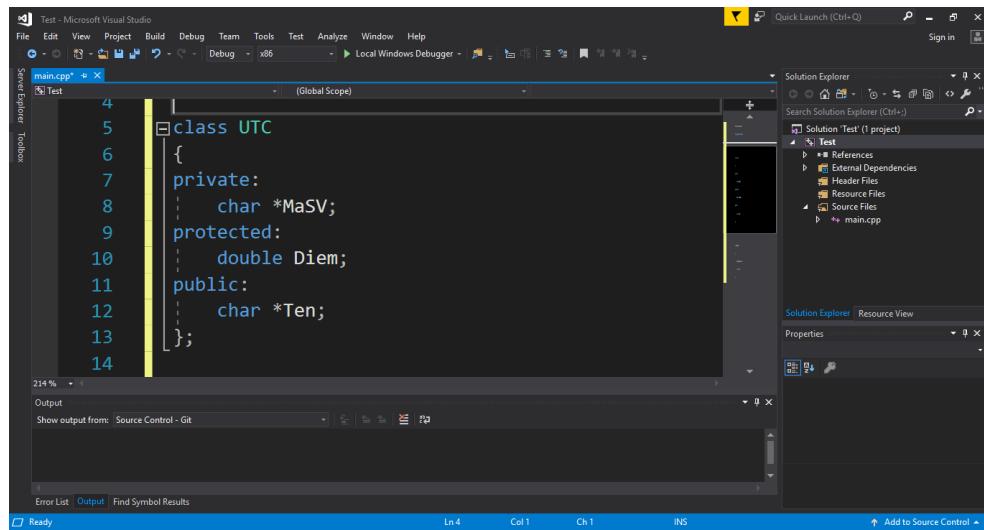
Trừu tượng hóa dữ liệu (*Data abstraction*) là một kỹ thuật lập trình mà dựa trên sự phân biệt của *Interface* (giao diện) và *Implementation* (trình triển khai).

Trong C++, chúng ta sử dụng các *class* để định nghĩa kiểu dữ liệu trừu tượng (*abstract data types (ADT)*) của riêng chúng ta.

c. Tính đóng gói

Tính đóng gói (*Encapsulation*) là một khái niệm của lập trình hướng đối tượng mà ràng buộc dữ liệu và các hàm mà thao tác dữ liệu đó, và giữ chúng an toàn bởi ngăn cản sự gây trở ngại và sự lạm dụng từ bên ngoài. Tính bao đóng dẫn đến khái niệm *OOP* quan trọng là *Data Hiding* (ẩn dữ liệu).

C++ hỗ trợ các thuộc tính của đóng gói và ẩn dữ liệu thông qua việc tạo các kiểu tự định nghĩa (user-defined), gọi là *classes*. Chúng ta đã học rằng một lớp có thể chứa các thành viên *private*, *protected* và *public*. Theo mặc định, tất cả thành phần được định nghĩa trong một lớp là *private*.



HÌNH 2. 6: *tính đóng gói*

d. Tính kế thừa

Một lớp có thể được kế thừa từ hơn một lớp khác, nghĩa là, nó có thể kế thừa dữ liệu và hàm từ nhiều lớp cơ sở. Để định nghĩa một lớp kế thừa (*Derived Class*), chúng ta sử dụng một danh sách để xác định các lớp cơ sở. Danh sách này liệt kê một hoặc nhiều lớp cơ sở và có *form* sau:

class lop_ke_thua: access_mode lop_co_so

access_modifier là *public*, *protected* hoặc *private*, và *lop_co_so* là tên của lớp đã được định nghĩa trước đó. Nếu *access_modifier* không được sử dụng, thì mặc định là *private*.

e. Tính đa hình

Đa hình là một hàm có nhiều hình thức thể hiện khác nhau tùy từng hoàn cảnh cụ thể.

Đa hình là một đặc trưng của ngôn ngữ lập trình hướng đối tượng.

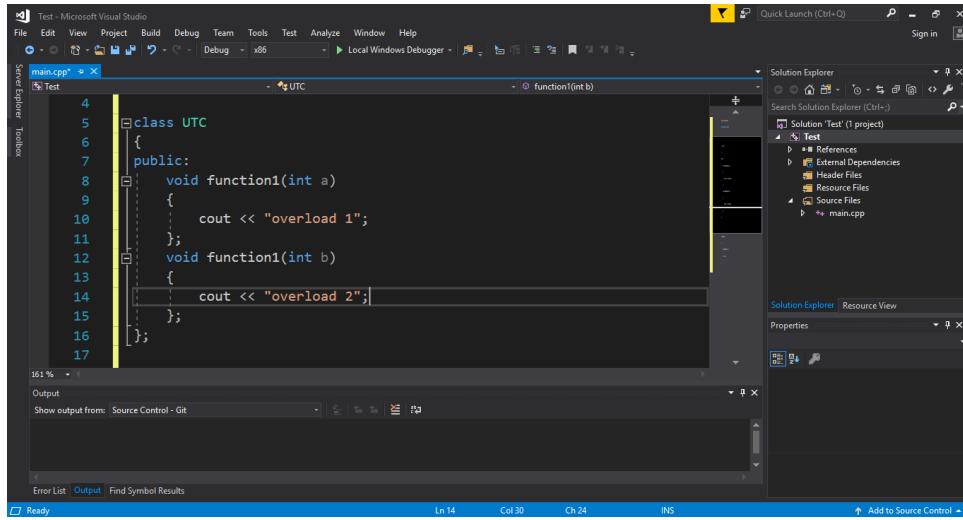
Có 2 loại đa hình: Đa hình tĩnh và đa hình động.

Trong đa hình tĩnh có 2 phương thức quan trọng đó là: Ghi đè (*Overriding*) và nạp chồng hàm (*Overloading*).

Overload một phương thức, tức là tạo ra nhiều phương thức có cùng tên nhưng khác nhau về danh sách tham số.

Overrid một phương thức, tức là tạo ra một phương thức trong lớp dẫn xuất có cùng prototype với một phương thức trong lớp cơ sở.

Nếu lớp cơ sở có một phương thức bị chèn và lớp dẫn xuất lại *override* phương thức này, thì phương thức của lớp dẫn xuất sẽ ẩn tất cả các phương thức của lớp cơ sở có cùng tên với nó.

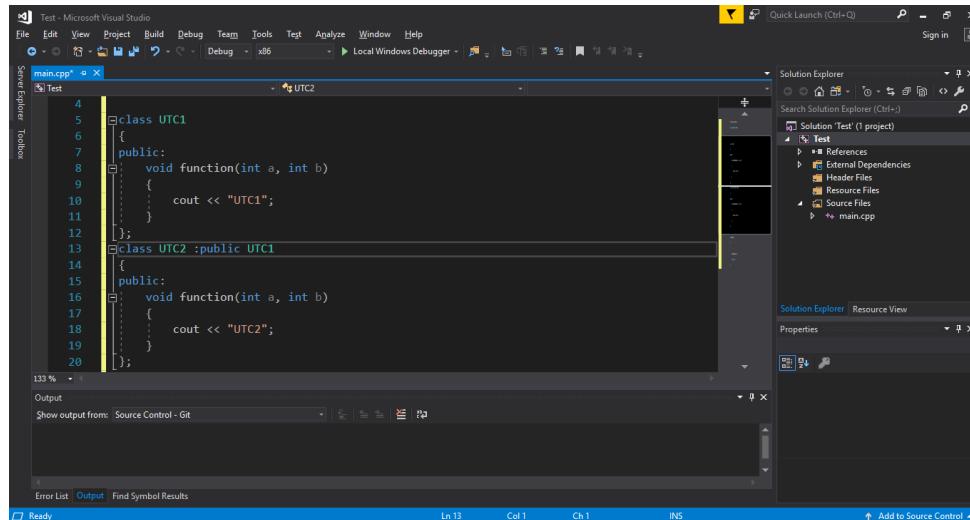


```

4
5     class UTC
6     {
7         public:
8             void function1(int a)
9             {
10                 cout << "overload 1";
11             }
12             void function1(int b)
13             {
14                 cout << "overload 2";
15             }
16     };
17

```

HÌNH 2. 7: đa hình tĩnh overloading



```

4
5     class UTC1
6     {
7         public:
8             void function(int a, int b)
9             {
10                 cout << "UTC1";
11             }
12     };
13     class UTC2 : public UTC1
14     {
15         public:
16             void function(int a, int b)
17             {
18                 cout << "UTC2";
19             }
20     };

```

HÌNH 2. 8: đa hình tĩnh overriding

Được thể hiện thông qua hàm ảo. Từ khoá *virtual* xác định hàm thành phần của lớp cơ sở sẽ bị override bởi lớp dẫn xuất. Khi lớp cơ sở định nghĩa các hàm ảo thì C++ sẽ tìm kiếm hàm đó trong lớp dẫn xuất trước, sau đó mới đến lớp cơ sở.

Hàm ảo

Lớp *Parent* và *Child* cùng có phương thức f

Khai báo một con trỏ thuộc kiểu của lớp Parent

Parent* p;

Con trỏ này trỏ đến đối tượng của lớp *Child*

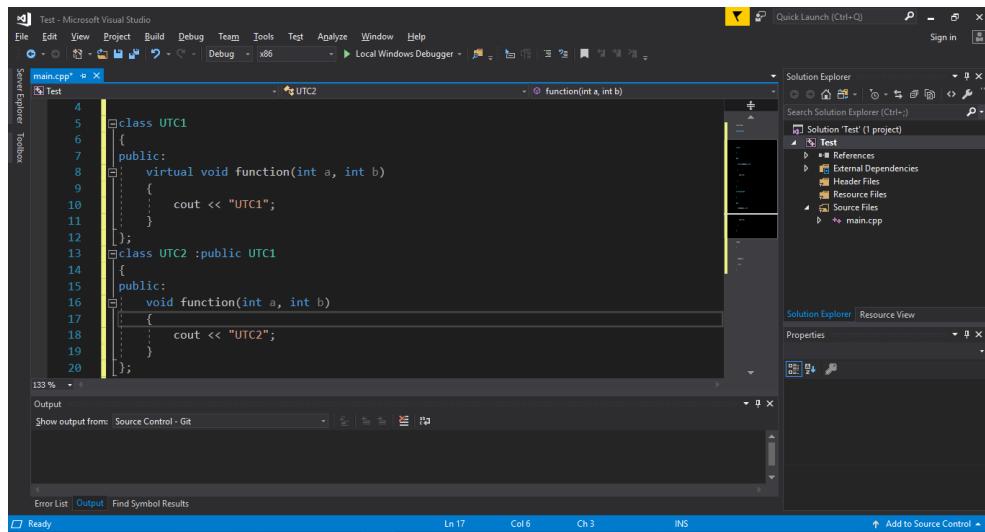
p = **new** Child;

Sau đó, thực hiện lời gọi

p ->f;

Kết quả: f của lớp Parent sẽ được viễn dẫn

Nếu f được khai báo là hàm ảo trong lớp *Parent* thì f của lớp *Child* sẽ được viễn dẫn.



```

4
5
6
7
8     class UTC1
9     {
10    public:
11        virtual void function(int a, int b)
12        {
13            cout << "UTC1";
14        }
15    };
16
17    class UTC2 :public UTC1
18    {
19    public:
20        void function(int a, int b)
21        {
22            cout << "UTC2";
23        }
24    };

```

HÌNH 2. 9: đa hình động

2.1.2 Lập trình Windows Destop Application

2.1.2.1 Tạo một project

Khởi động visual studio

Trên Menu: *File* → *New* → *Project* (*Ctrl* + *Shift* + *N*)

Các thao tác:

- Chọn ngôn ngữ lập trình Visual C++.
- Chọn *Windows Destop Application*.
- Đặt tên project trong mục *Name*.

- Chọn đường dẫn lưu *project* trong *Location*.
 - Đặt tên dự án quản lý trong *Solution name*.

Click chọn *OK* để bắt đầu tạo *project*.

Click tên *project* trong *solution*, chuột phải chọn *Add* → *New Item*.

Ở đây ta có thể chọn *add *.cpp, *.h, class*.

2.1.2.2 Cấu trúc chương trình trong windows

Mỗi chương trình ứng dụng trong Windows bắt buộc phải có 2 hàm:

- *WinMain()*
 - *Window procedure*

a. Hàm *WinMain()*

Tương tự như hàm `main()` (trong *Dos* hoặc *UNIX*) khởi tạo chương trình ứng dụng, dùng để: hiển thị cửa sổ ứng dụng lên màn hình, tiến hành vòng lặp *message*.

Khai báo hàm *WinMain()*:

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,LPSTR  
lpCmdLine, int nCmdShow );
```

hInstance: là một thể hiện của chương trình. Nó là số nguyên 32bit, số nguyên này sẽ được cho bởi Windows khi chương trình ứng dụng bắt đầu thực hiện.

hPrevInstance : là thông số để *NULL*

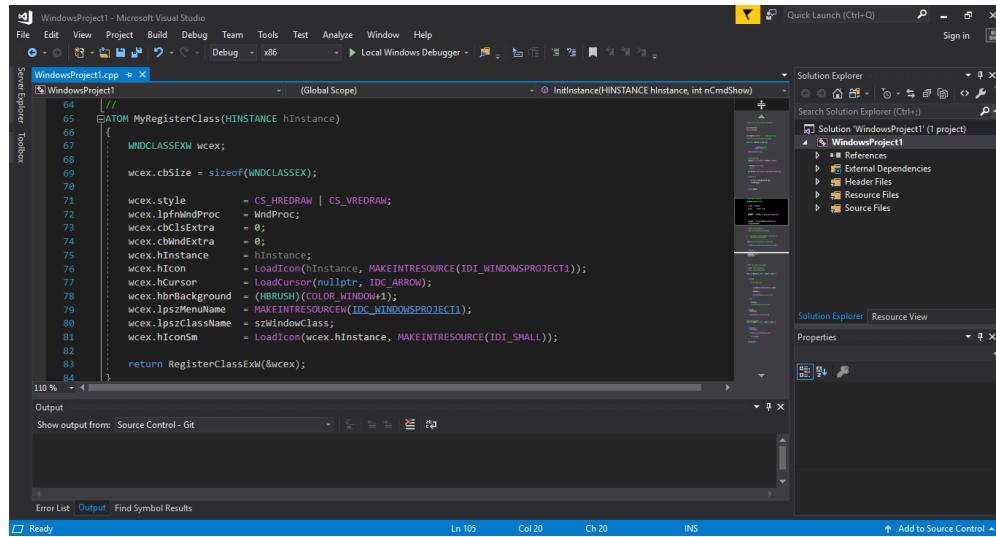
lpCmdLine:

nCmdShow: chỉ ra cửa sổ sẽ được hiển thị như thế nào (*Minimized, maximized, Hidden*).

Hàm *WinMain()* sẽ kết thúc khi nó nhận được bản tin *WM_QUIT*.

Hàm *WinMain()* được tạo theo các bước sau:

- Đăng ký cửa sổ
 - Tạo cửa sổ
 - Tao vòng lặp *message*

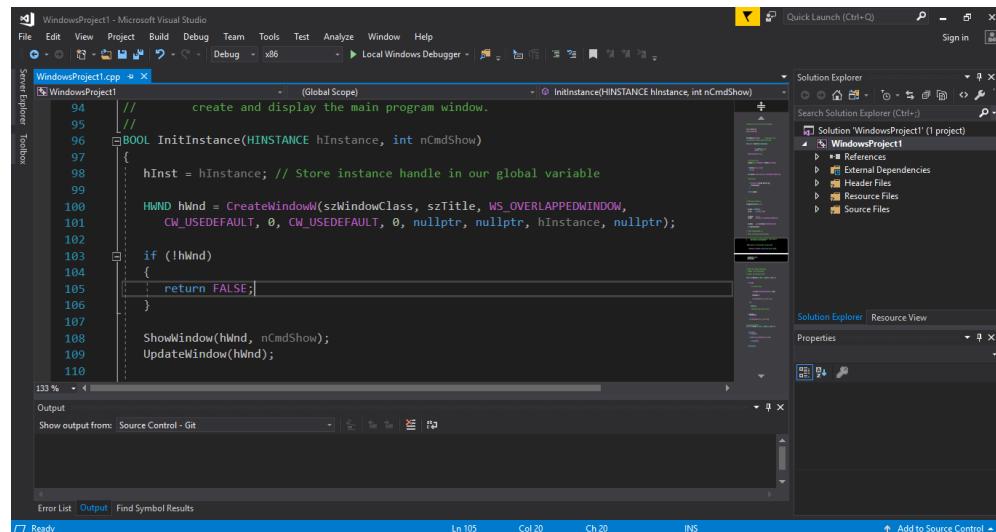


```

WindowsProject1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Local Windows Debugger - x86 - Local Windows Debugger - 
Solution Explorer
WindowsProject1
WindowsProject1.cpp
64 // EATON MyRegisterClass(HINSTANCE hInstance)
65 {
66     WNDCLASSEXW wcex;
67
68     wcex.cbSize = sizeof(WNDCLASSEX);
69
70     wcex.style = CS_HREDRAW | CS_VREDRAW;
71     wcex.lpfnWndProc = WndProc;
72     wcex.cbClsExtra = 0;
73     wcex.cbWndExtra = 0;
74     wcex.hInstance = hInstance;
75     wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_WINDOWSPROJECT1));
76     wcex.hCursor = LoadCursor(nullptr, IDC_ARROW);
77     wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
78     wcex.lpszMenuName = MAKEINTRESOURCE(IDC_WINDOWSPROJECT1);
79     wcex.lpszClassName = szWindowClass;
80     wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));
81
82     return RegisterClassExW(&wcex);
83 }
84

```

HÌNH 2. 10: đăng ký cửa sổ



```

WindowsProject1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Local Windows Debugger - x86 - Local Windows Debugger - 
Solution Explorer
WindowsProject1
WindowsProject1.cpp
94 // create and display the main program window.
95 //
96 BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
97 {
98     hInst = hInstance; // Store instance handle in our global variable
99
100     HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
101                             CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, nullptr, hInstance, nullptr);
102
103     if (!hWnd)
104     {
105         return FALSE;
106     }
107
108     ShowWindow(hWnd, nCmdShow);
109     UpdateWindow(hWnd);
110

```

HÌNH 2. 11: tạo cửa sổ

```

WindowsProject1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
WindowsProject1.cpp - (Global Scope) Local Windows Debugger
MSG msg;
// Main message loop:
while (GetMessage(&msg, nullptr, 0, 0))
{
    if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
return (int) msg.wParam;
// FUNCTION: MyRegisterClass()
// PURPOSE: Registers the window class.

```

HÌNH 2. 12: vòng lặp message

b. Hàm *Window procedure*

LRESULT CALLBACK *WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);*

Hàm *WindowProc()* sẽ nhận và xử lí các *message* gửi đến. Các *message* nào không được xử lí trong hàm này sẽ được xử lí trong hàm *DefWindowProc()* của *HĐH Windows*.

hwnd: handle to the *windows*

uMsg

wParam, lParam: chứa các thông tin về *message*

c. Các *message* cơ bản

WM_CHAR	Khi nhập 1 kí tự từ bàn phím
WM_COMMAND	Khi lựa chọn các item trong popup menu
WM_CREATE	Khi windows được tạo
WM_DESTROY	Khi windows bị destroy
WM_LBUTTONDOWN	Khi click chuột trái
WM_RBUTTONDOWN	Khi click chuột phải
WM_MOUSEMOVE	Khi di chuyển con trỏ chuột
WM_PAINT	Khi windows được vẽ lại
WM_QUIT	Khi close windows

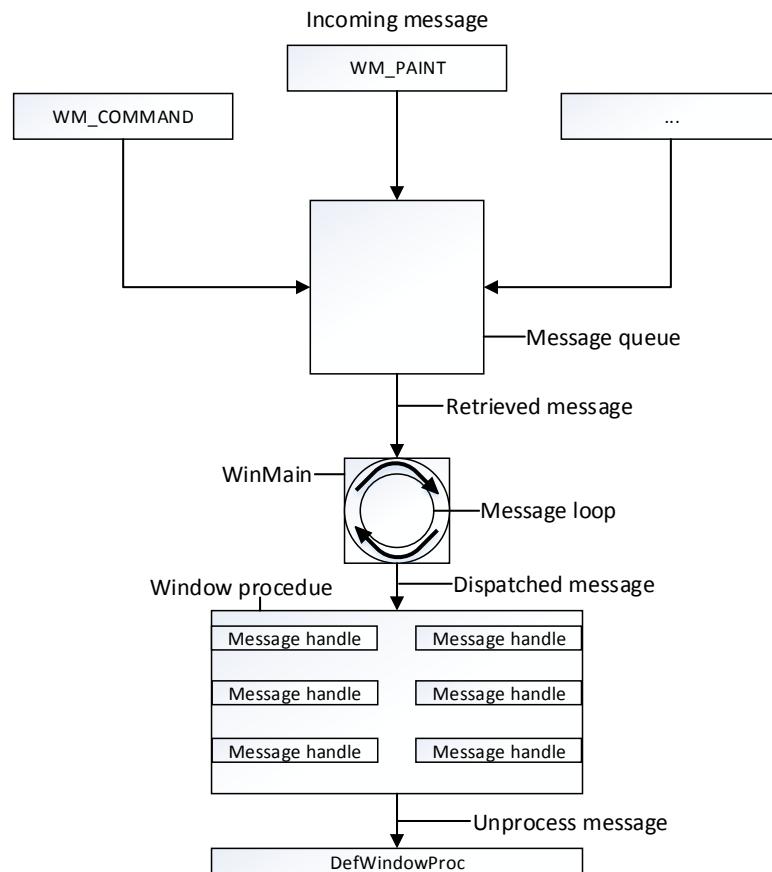
BẢNG 2. 8: các *message* cơ bản

Trong windows tất cả những nút bấm(button), editbox (hộp soạn thảo), listbox, combobox,...đều được coi là một *windows*(cửa sổ), chính vì thế khi làm việc trong

Windows bạn hãy coi mỗi thành phần chúng ta làm là một cửa sổ để điều hành và quản lý chúng. Mỗi cửa sổ Windows (như nói ở trên) có một *handle*, *handle* là giá trị chứa quyền quản lý Windows đó, khi bạn có một button hay một *static text*, ... bạn sẽ có một *handle* của nó, và bạn có thể điều khiển (thay đổi nội dung, làm cho nhìn thấy được, ẩn đi,...) những thành phần này.

Cốt lõi của Windows là việc xử lý thông điệp, thông điệp là gì, thông điệp là những thứ ta gửi qua nó để điều khiển Windows, ví dụ chúng ta bấm chuột vào Windows thì sự kiện bấm chuột đó được gửi tới Windows thông qua thông điệp *WM_LBUTTONDOWN*, thông điệp này sẽ nằm trong hàng đợi và đợi Windows xử lý, việc nằm trong hàng đợi trong khoảng thời gian ngắn nên bạn không thể xác định được, và cửa sổ của bạn lại đợi tiếp những thông điệp khác đến khi chương trình bạn kết thúc.

Quá trình nhận và xử lý *message*:



HÌNH 2. 13: sơ đồ xử lý message

CHƯƠNG 3 : THIẾT KẾ, LẬP TRÌNH ỨNG DỤNG

3.1 PHÂN TÍCH YÊU CẦU

3.1.1 Mô tả tổng quan

3.1.1.1 Các chức năng của chương trình

- Tra cứu từ điển tiếng anh và tra cứu từ điển tiếng việt.
 - Thêm mới từ và dữ liệu của ứng dụng.
 - Sửa lại nghĩa của từ theo yêu cầu của người sử dụng.
 - Xóa từ tạm thời và xóa từ khỏi dữ liệu của ứng dụng.

3.1.1.2 Sử dụng case diagram

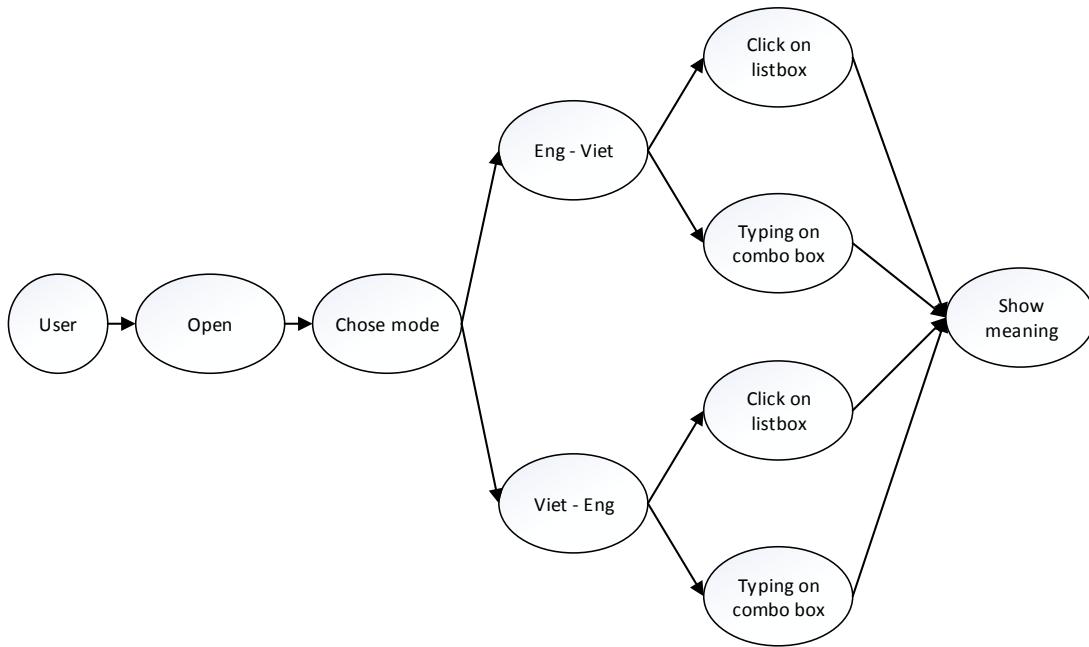
#	Code	Name	Description
1	UC01	Search dictionary	Tra từ, tìm kiếm nghĩa tiếng anh và tiếng việt.
2	UC02	Add word	Thêm một từ mới hoặc một từ đã có sẵn thì chuyển sang chế độ update nghĩa.
3	UC03	Delete word	Xóa từ trong từ điển.
4	UC04	Edit word	Chỉnh sửa nghĩa của từ.

BẢNG 3. 1: các case diagram

3.1.1.3 Mô tả diagram

3.1.1.3.1 Use case 01 (UC01) – Search dictionary

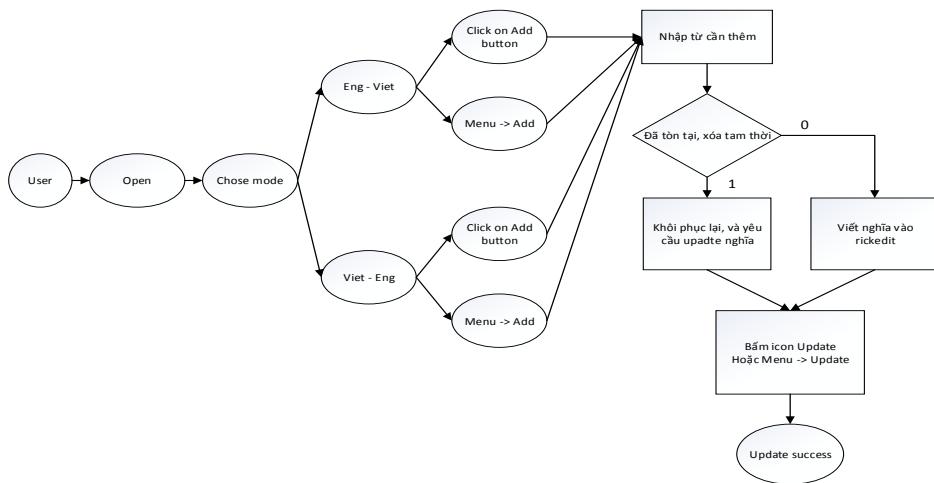
User mở chương trình từ điển, chọn từ điển muốn sử dụng (Anh – Việt hoặc Việt - Anh). Tìm kiếm từ bằng cách gõ trên *combobox* rồi bấm vào button tìm kiếm, hoặc *click* vào từ trên *list box*, EVDict sẽ hiển thị nghĩa sau thao tác của người dùng trên *richedit*.



HÌNH 3. 1: Use case 01: Search dictionary

3.1.1.3.2 Use case 02 (UC02) – Add word

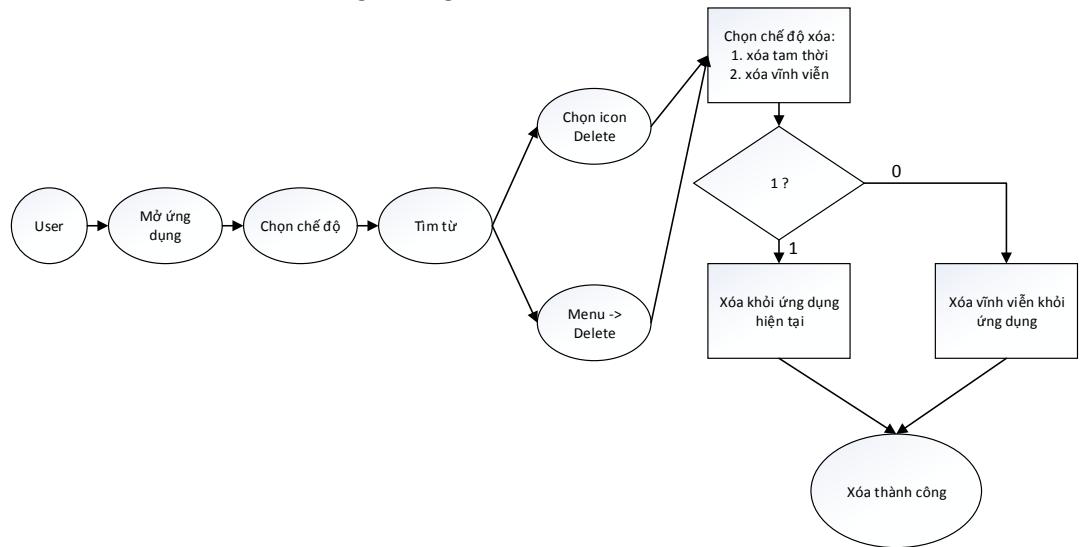
User mở chương trình từ điển, chọn từ điển muốn sử dụng (Anh–Việt hoặc Việt - Anh). Thêm từ bằng cách ấn nút Add trên từ điển hoặc chọn *Menu* → *Add*. Ứng dụng sẽ hiện thị một cửa sổ, nhập từ cần thêm vào. Nếu từ đó không trùng với bất kỳ từ nào có trong từ điển, và không trùng với từ đã bị xóa tạm thời. Thì sẽ có thông báo yêu cầu nhập nghĩa vào richedit, sau khi nhập xong bấm vào icon *Update* hoặc chọn *Menu* → *Update*, thông báo cập nhật từ mới thành công. Nếu từ đó trùng với từ đã có trong danh sách từ điển hoặc từ đã xóa tạm thời thì từ đó được trả lại và yêu cầu *Update* nghĩa. Làm tương tự, thì sẽ thêm thành công.



HÌNH 3. 2: Use case 02: Add word

3.1.1.3.3 Use case 03 (UC03): Delete word

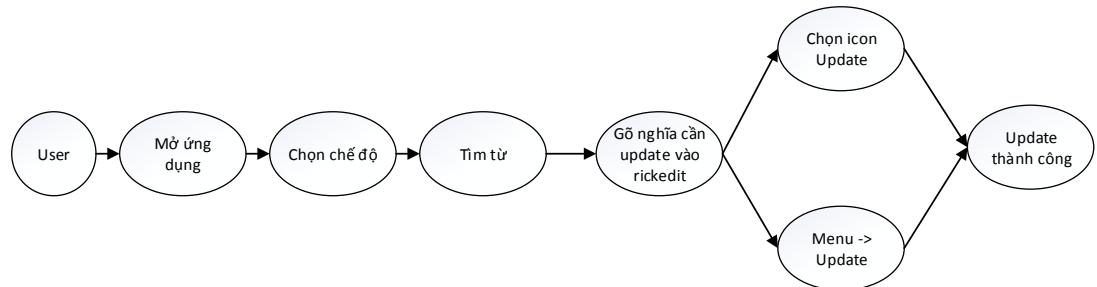
User mở chương trình từ điển, chọn từ điển muốn sử dụng. Tìm kiếm từ muốn xóa, ấn nút *Delete* trên *toolbar* hoặc chọn *Menu* → *Delete*. Từ điển sẽ hiện lên một cửa sổ, có các lựa chọn: “*Delete*”, “*Destroy*”, “*Cancel*”. Nếu chọn “*Delete*”, từ đó sẽ bị xóa tạm ra khỏi ứng dụng, sau khi mở lại ứng dụng, từ đó vẫn còn. Nếu chọn “*Destroy*”, từ đó sẽ bị xóa vĩnh viễn ra khỏi ứng dụng, mở lên từ đó không còn. Nếu chọn “*Cancel*”, thì không làm gì cả.



HÌNH 3. 3: Use case 03: Delete word

3.1.1.3.4 Use case 04 (UC04): Update word

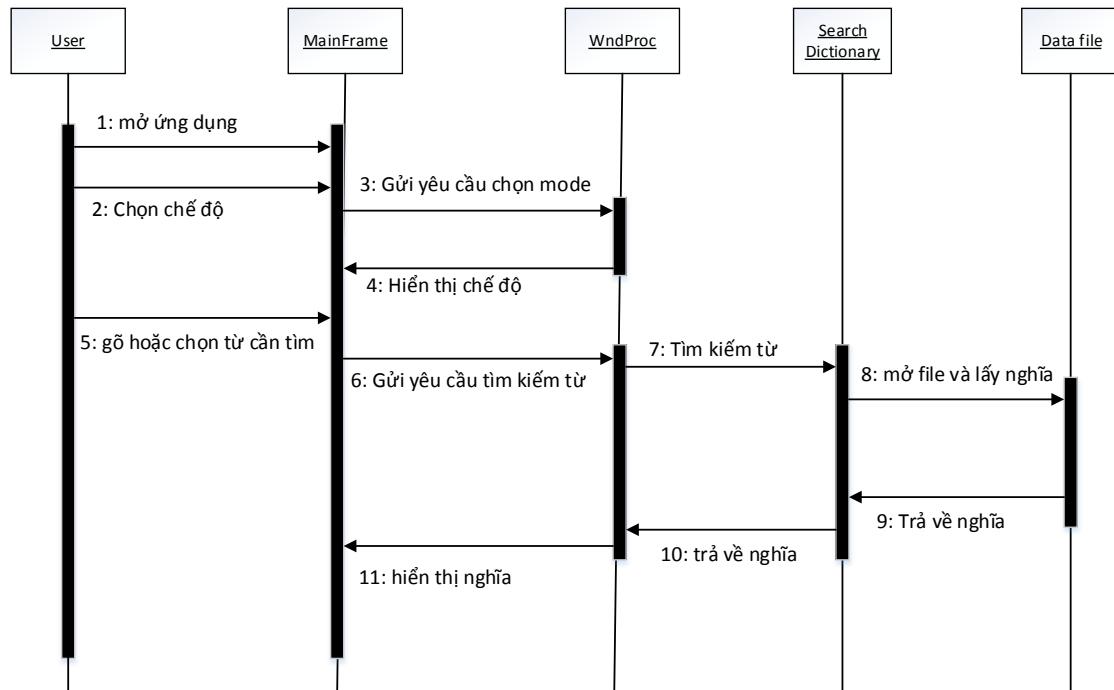
User mở chương trình từ điển *EVDict*, chọn từ điển muốn sử dụng. Tìm kiếm từ muốn *update*, gõ nghĩa mới vào *RichEdit*, chọn nút *Update* trên *toolbar* hoặc *Menu* → *Update*, từ điển sẽ thông báo “*Update success*” sau khi hoàn thành việc *update*.



HÌNH 3. 4: Use case 04: Update word

3.1.1.4 Mô tả luồng dữ liệu

3.1.1.4.1 Tìm kiếm từ



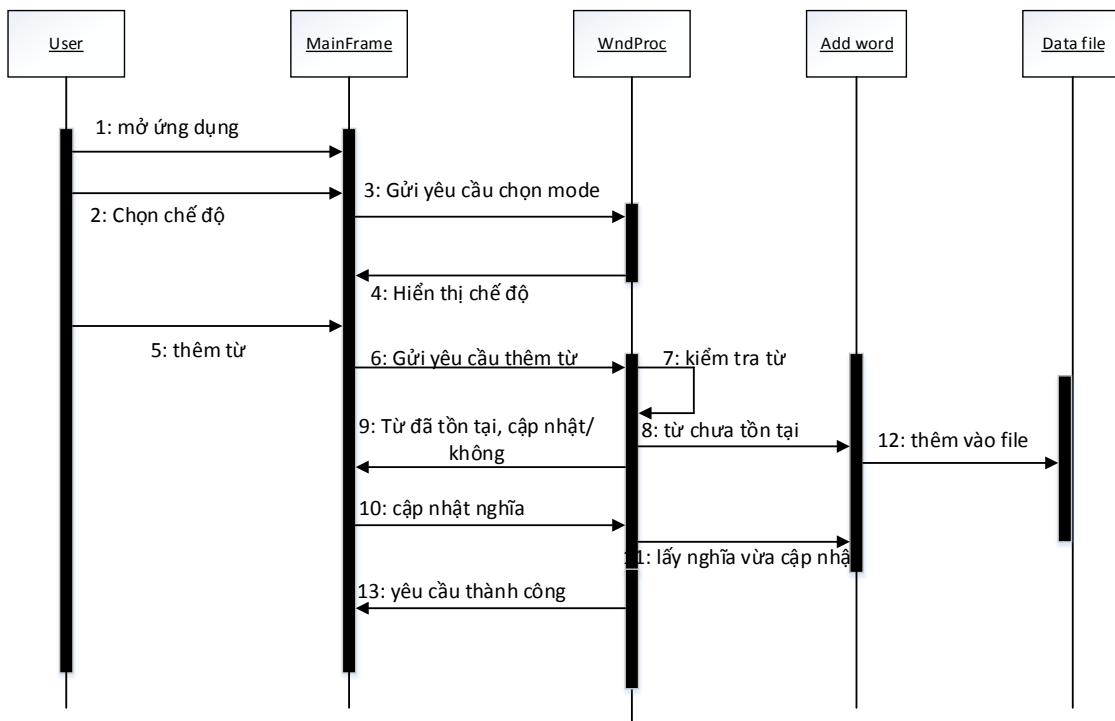
HÌNH 3. 5: sơ đồ luồng thực thi tìm kiếm

Mô tả trình tự thực hiện Use case 01 – Search word

Bước	Mô tả
1	Người dùng mở từ điển EVDict.
2	Người dùng chọn mode từ điển (mặc định khi khởi động chương trình là từ điển Anh – Việt), MainFrame gửi thông số mode tới hàm WndProc
3	MainFrame gửi yêu cầu thay đổi mode tới WndProc
4	WndProc gửi thông điệp hiển thị mode được chọn tới MainFrame.
5	User nhập từ cần tìm kiếm, hoặc chọn từ cần tìm ở richedit.
6	MainFrame gửi yêu cầu tìm kiếm nghĩa tới hàm WndProc.
7	WndProc tìm kiếm từ cần thiết.
8	Mở file dữ liệu và lấy ra dữ liệu cần thiết.
9	Trả về chuỗi nghĩa của từ.
10	Trả về chuỗi nghĩa của từ
11	Hiển thị nghĩa trên MainFrame.

BẢNG 3. 2: luồng thực thi tìm kiếm

3.1.1.4.2 Thêm từ



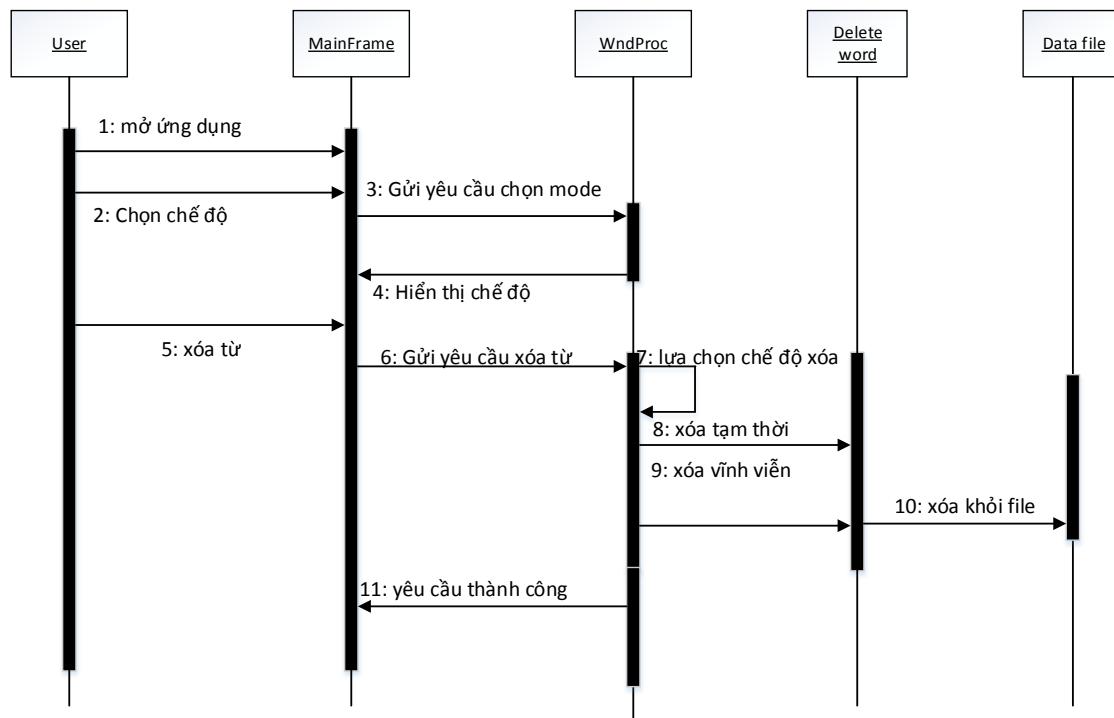
HÌNH 3. 6: sơ đồ luồng thực thi thêm từ

Mô tả trình tự thực hiện Use case 02 – Add word

Bước	Mô tả
1	Người dùng mở từ điển EVDict.
2	Người dùng chọn mode từ điển (mặc định khi khởi động chương trình là từ điển Anh – Việt), MainFrame gửi thông số mode tới hàm WndProc.
3	MainFrame gửi yêu cầu thay đổi mode tới WndProc
4	WndProc gửi thông điệp hiển thị mode được chọn tới MainFrame.
5	User bấm nút thêm từ, nhập từ cần thêm.
6	Gửi yêu cầu thêm từ tới hàm WndProc.
7	WinProc thực hiện việc kiểm tra sự tồn tại của từ.
8	Xác định từ chưa tồn tại
9	WndProc gửi thông báo tới người dùng khi từ đã tồn tại.
10	User đồng ý update nghĩa.
11	WndProc gọi hàm để lấy nghĩa của từ vừa nhập ở richedit.
12	Thêm dữ liệu (từ và nghĩa) vào file dữ liệu.
13	Gửi thông báo thêm từ thành công.

BẢNG 3. 3: luồng thực thi thêm từ

3.1.1.4.3 Xóa từ



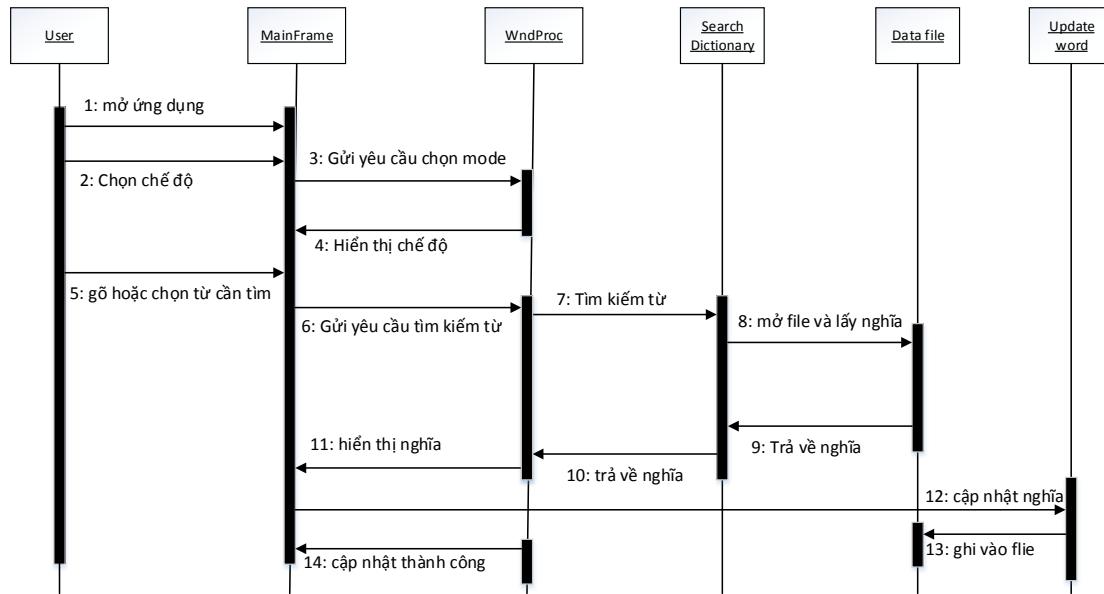
HÌNH 3. 7: sơ đồ luồng thực thi xóa từ

Mô tả trình tự thực hiện Use case 03 – Delete word

Bước	Mô tả
1	Người dùng mở từ điển EVDict.
2	Người dùng chọn mode từ điển (mặc định khi khởi động chương trình là từ điển Anh – Việt).
3	Gửi yêu cầu thay đổi chế độ từ điển tới WndProc.
4	WndProc trả về giá trị của mode (1 là từ điển Anh – Việt, 2 là từ điển Việt - Anh)
5	User tìm kiếm từ rồi chọn xóa.
6	Gửi yêu cầu xóa từ tới WndProc .
7	Lựa chọn chế độ xóa: xóa tạm thời , xóa vĩnh viễn
8	Xóa tạm thời từ đó ra khỏi ứng dụng, khi khởi động lại từ đó được khôi phục
9	Xóa vĩnh viễn từ đó ra khỏi dữ liệu của ứng dụng.
10	Xóa từ khỏi file
11	Hiện thông báo xóa từ thành công.

BẢNG 3. 4: luồng thực thi xóa từ

3.1.1.4.4 Cập nhật nghĩa của từ



HÌNH 3. 8: sơ đồ luồng thực thi cập nhật nghĩa của từ

Mô tả trình tự thực hiện Use case 04 – Update word

Bước	Mô tả
1	Người dùng mở từ điển EVDict.
2	Người dùng chọn mode từ điển (mặc định khi khởi động chương trình là từ điển Anh – Việt), MainFrame gửi thông số mode tới hàm WndProc.
3	Gửi yêu cầu thay đổi chế độ từ điển tới WndProc.
4	WndProc trả về giá trị của mode (1 là từ điển Anh – Việt, 2 là từ điển Việt - Anh)
5	Tìm kiếm từ cần cập nhật.
6	Gửi yêu cầu tìm kiếm từ.
7	Tìm kiếm từ cần cập nhật.
8	mở file lấy nghĩa
9	Trả về nghĩa của từ vừa tìm thấy
10	Trả về nghĩa của từ vừa tìm thấy
11	Hiển thị nghĩa
12	Chỉnh sửa nghĩa của từ
13	Ghi vào file
14	Gửi thông báo update thành công.

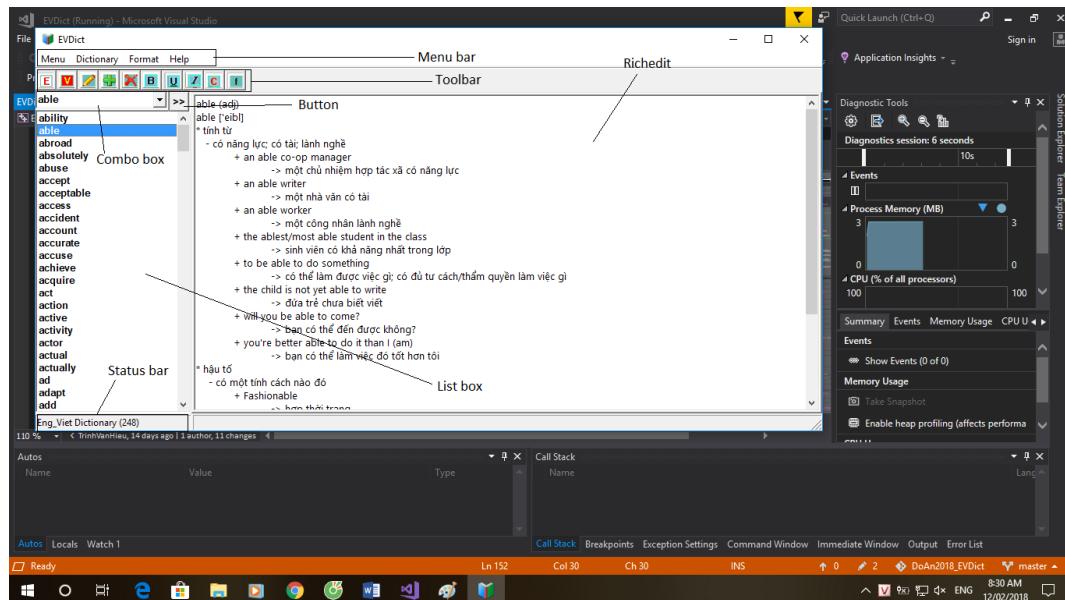
BẢNG 3. 5: luồng thực thi cập nhật từ

3.1.2 Thiết kế

3.1.2.1 Giao diện hiển thị

Các thành phần chính của giao diện *EVDict* bao gồm:

- *Menu bar*
- *Toolbar*
- *Combobox*
- *List box*
- *Richedit*
- *Status bar*



HÌNH 3. 9: giao diện chính của ứng dụng *EVDict*

3.1.2.2 Mô tả *Menu bar*

Menu bar bao gồm: *Menu, Dictionary, Format, Help*

Menu bao gồm các chức năng:

- *Update*: cập nhật nghĩa của từ.
- *Add*: thêm từ mới cho từ điển.
- *Delete*: xóa từ.
- *Exit*: thoát chương trình

Dictionary bao gồm 2 chế độ:

- Eng – Viet: chế độ tra cứu từ điển Anh – Việt.
 - Viet – Eng: chế độ tra cứu từ điển Việt – Anh

Format:

- *Font*: thay đổi chế độ font chữ hiển thị.

Help:

- *About*: thông tin về người lập trình ra ứng dụng này.

3.1.2.3 Mô tả *Toolbar*

Toolbar gồm 9 icon:

- *Icon* : chế độ tra cứu từ điển Anh – Việt.
 - *Icon* : chế độ tra cứu từ điển Việt – Anh.
 - *Icon* : chức năng cập nhật nghĩa của từ.
 - *Icon* : chức năng thêm mới từ.
 - *Icon* : chức năng xóa từ trong từ điển.
 - *Icon* : chức năng chọn/ bỏ chọn chế độ in đậm cho từ và nghĩa.
 - *Icon* : chức năng chọn/ bỏ chọn chế độ gạch chân cho từ và nghĩa.
 - *Icon* : chức năng chọn/ bỏ chọn chế độ in nghiêng cho từ và nghĩa.
 - *Icon* : chức năng chọn/ bỏ chế độ màu sắc cho từ và nghĩa.
 - *Icon* : chức năng chọn/ bỏ định dạng font chữ cho từ và nghĩa.

3.1.2.4 Mô tả *Combobox* và *Button*

Combobox dùng để gõ từ cần tìm kiếm. *Combobox* có khả năng lưu những từ đã tìm kiếm gần thời điểm hiện tại nhất.

Button: dùng để thực hiện thao tác tìm kiếm sau khi được ấn, nếu tìm thấy thì từ đó được lưu lại trong lịch sử của *Combobox*.

3.1.2.5 Mô tả *Listbox*

Listbox hiển thị toàn bộ danh sách từ có trong ứng dụng tra cứu từ điển.

3.1.2.6 Mô tả *Richedit*

RichEdit hiển thị từ và nghĩa của từ, có thể thay đổi định dạng trên *richedit* như: *font* chữ, in đậm, gạch chân, in nghiêng, màu sắc. Ngoài ra việc *update* nghĩa mới cho từ được thực hiện trên *richedit*.

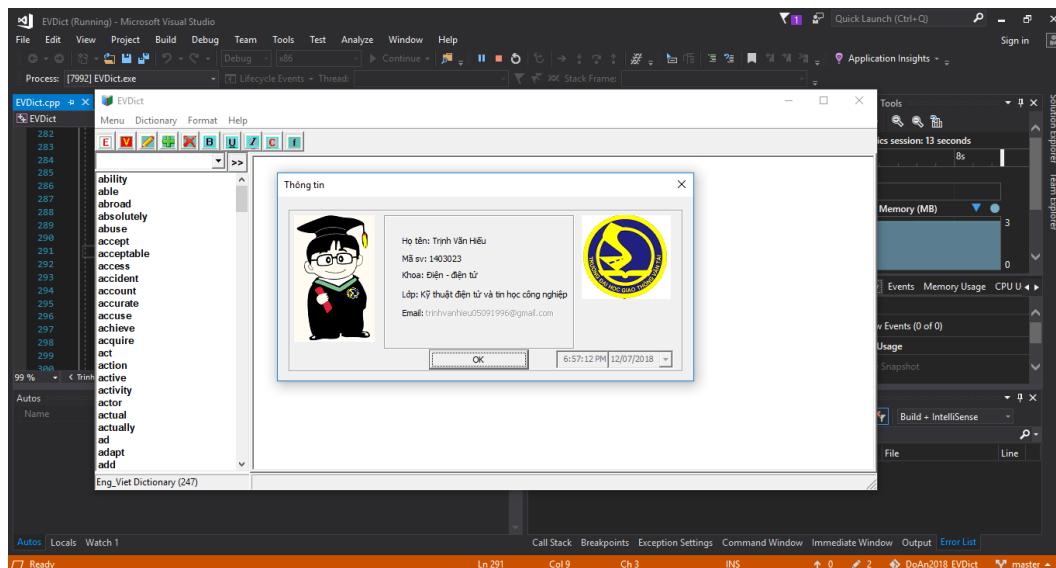
3.1.2.7 Mô tả *Status bar*

Status bar hiển thị chế độ từ điển đang bật của chương trình, đồng thời cũng hiển thị số lượng từ đang hiển thị trong *Listbox*.

3.1.2.8 Mô tả *Dialog box*

a. *Dialog About*

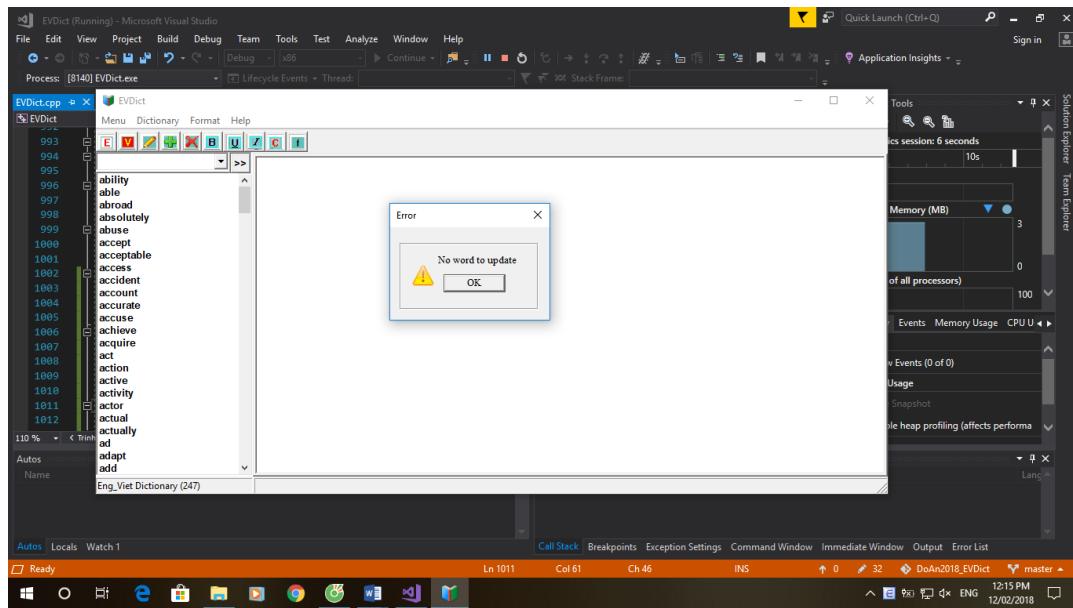
Hộp thoại *About* hiện lên khi mở ứng dụng hoặc người dùng chọn vào tính năng *About* của *Help* trên *Menu bar*.



HÌNH 3. 10: hộp thoại thông tin

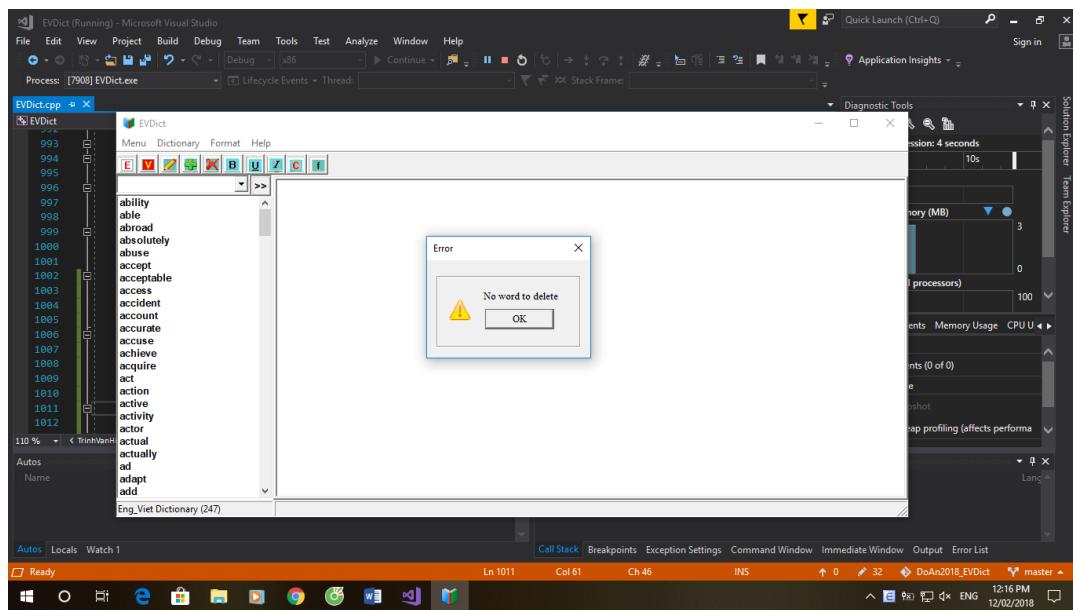
b. *Dialog Error*

Hộp thoại *Update error* hiện lên khi *user* chọn chức năng *update* mà chưa có từ nào để chọn.



HÌNH 3. 11: Update Error

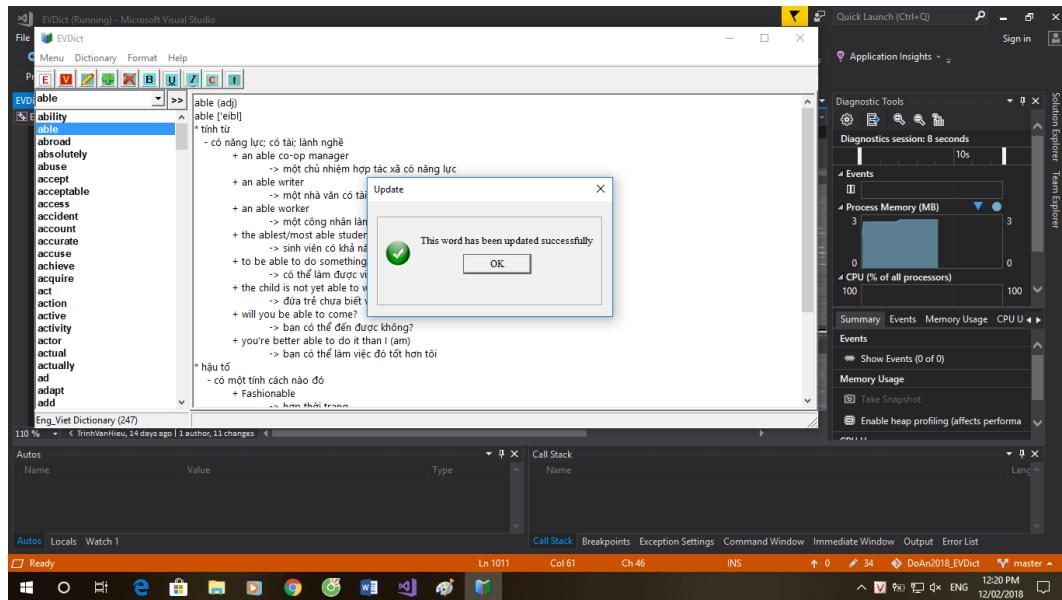
Hộp thoại *Delete error* hiện lên, khi người dùng chọn chức năng *delete* mà chưa có từ nào để chọn.



HÌNH 3. 12: Delete error

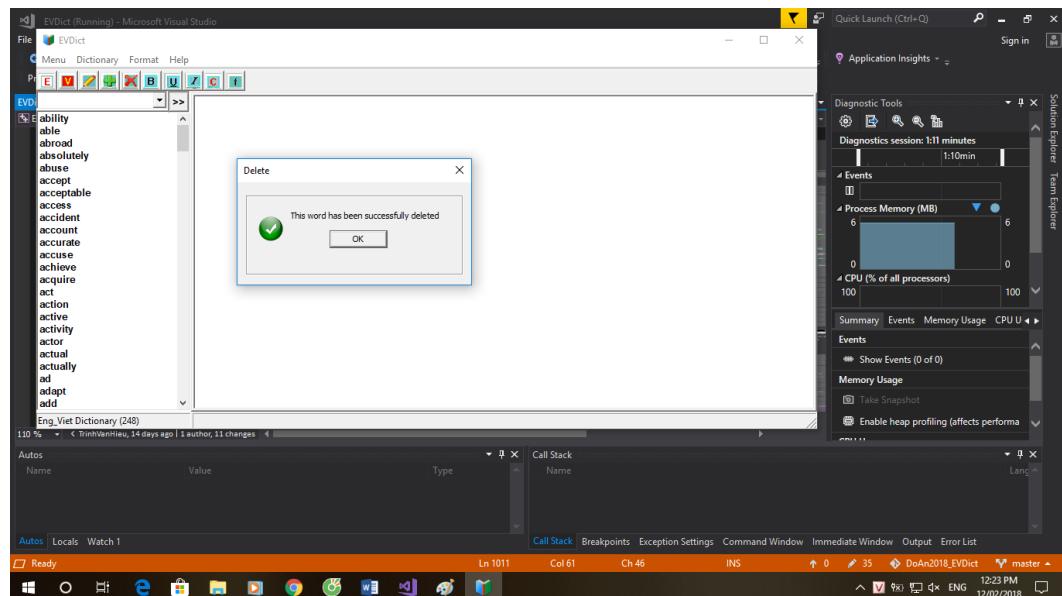
c. *Dialog success*

Hộp thoại *Update success* hiện lên, khi người dùng *update* nghĩa thành công của từ.



HÌNH 3. 13: *Update success*

Hộp thoại *Delete success* hiện lên, khi người dùng vừa xóa thành công một từ.

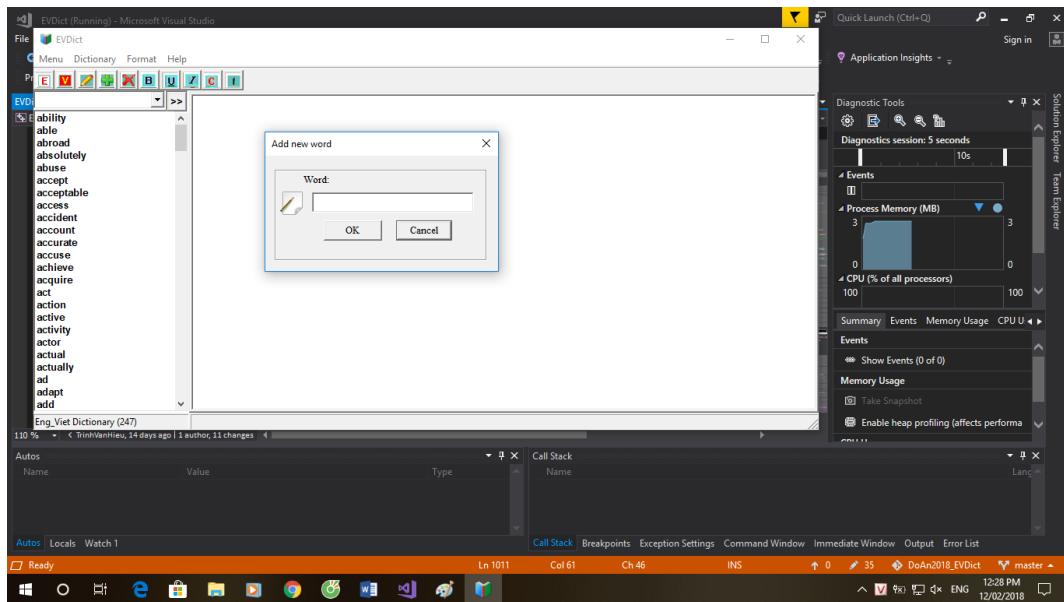


HÌNH 3. 14: Delete success

d. *Dialog Add*

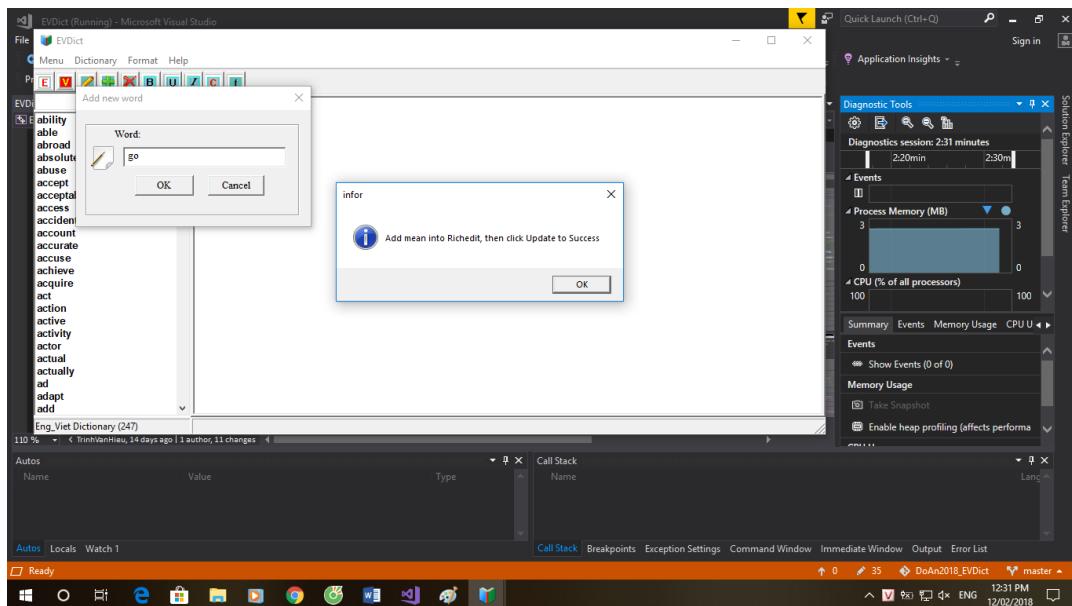
Hộp thoại *Add* hiện lên, khi người dùng bấm vào icon *Add* trên *Toolbar* hoặc

Menu → Add.



HÌNH 3. 15: Add new word

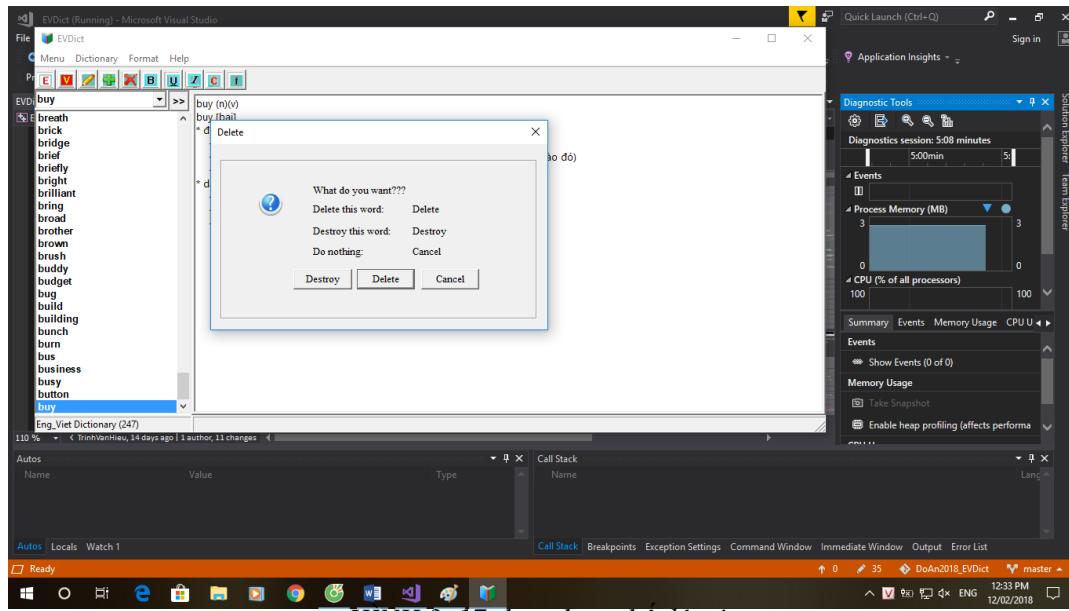
Sau khi thêm 1 từ, sẽ có thông báo yêu cầu nhập nghĩa của từ vừa rồi lên *richedit*, sau đó bấm *Update* để hoàn thành quá trình thêm từ.



HÌNH 3. 16: Add mean

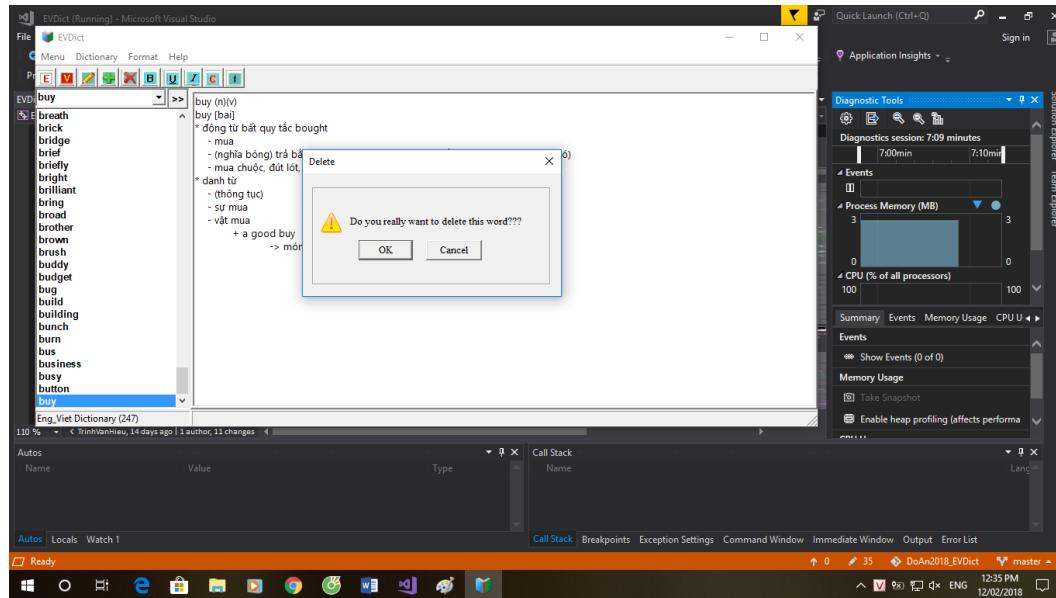
d. Dialog Delete

Sau khi chọn 1 từ và ấn vào icon *Delete* trên *Toolbar*, hoặc *Menu* → *Delete*.
Thì hộp thoại lựa chọn chế độ xóa được hiện ra.

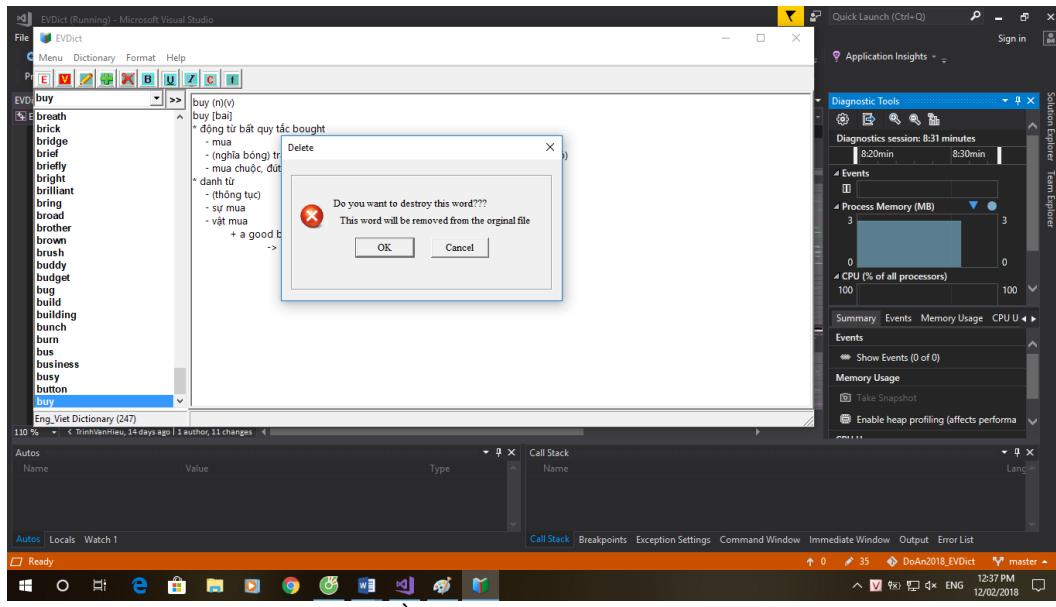


HÌNH 3. 17: lựa chọn chế độ xóa

Chọn chế độ, và hộp thoại thông báo và cảnh báo sẽ được hiện lên.



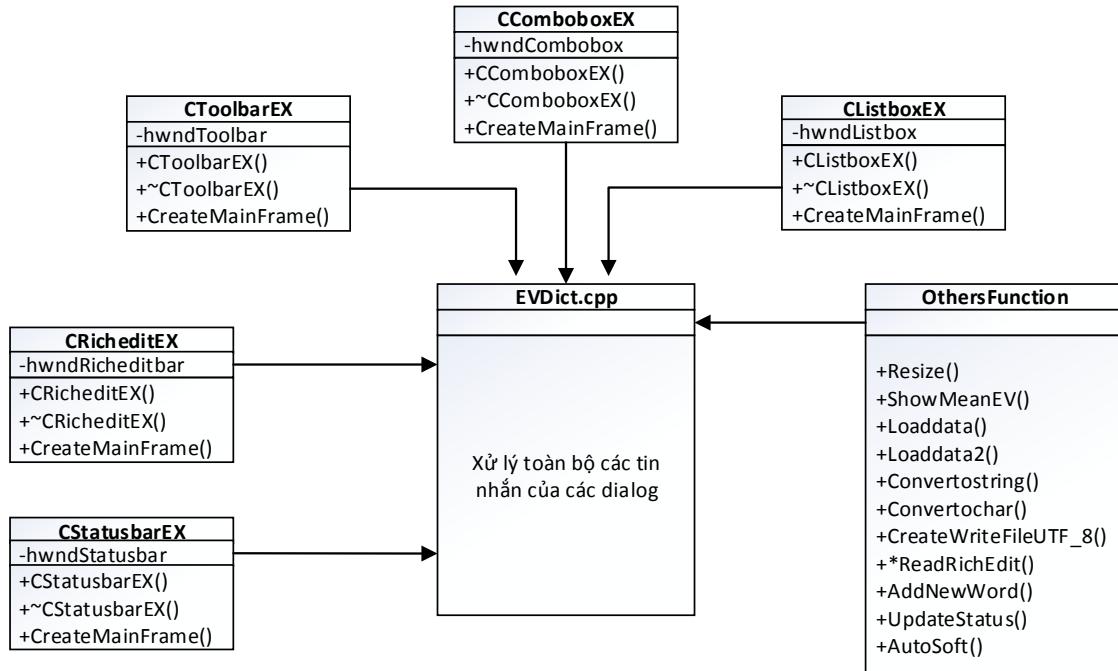
HÌNH 3. 18: Delete word



HÌNH 3. 19: Destroy word

3.2 LẬP TRÌNH

3.2.1 Class diagram



HÌNH 3. 20: class diagram

Mô tả các *class diagram*:

#	Tên	Loại	Kiểu	Mô tả
1	hwndCombobox	property	HWND	Handle của một cửa sổ Combobox
2	CComboboxEX()	method	void	Hàm khởi tạo cửa sổ Combobox không đối số
3	~CComboboxEX()	method	void	Hàm hủy cửa sổ Combobox không đối số
4	CreateMainFrame()	method	HWND	Hàm tạo cửa sổ Combobox có đối số

#	Tên	Loại	Kiểu	Mô tả
1	hwndListbox	property	HWND	Handle của một cửa sổ Listbox
2	CListboxEX()	method	void	Hàm khởi tạo cửa sổ Listbox không đối số
3	~CListboxEX()	method	void	Hàm hủy cửa sổ Listbox không đối số
4	CreateMainFrame()	method	HWND	Hàm tạo cửa sổ Listbox có đối số

#	Tên	Loại	Kiểu	Mô tả
1	hwndRichedit	property	HWND	Handle của một cửa sổ Richedit
2	CRicheditEX()	method	void	Hàm khởi tạo cửa sổ Richedit không đối số
3	~CRicheditEX()	method	void	Hàm hủy cửa sổ Richedit không đối số
4	CreateMainFrame()	method	HWND	Hàm tạo cửa sổ Richedit có đối số

#	Tên	Loại	Kiểu	Mô tả
1	hwndToolbar	property	HWND	Handle của một cửa sổ Toolbar
2	CToolbarEX()	method	void	Hàm khởi tạo cửa sổ Toolbar không đổi số
3	~CToolbarEX()	method	void	Hàm hủy cửa sổ Toolbar không đổi số
4	CreateMainFrame()	method	HWND	Hàm tạo cửa sổ Toolbar có đổi số

#	Tên	Loại	Kiểu	Mô tả
1	hwndStatusbar	property	HWND	Handle của một cửa sổ Statusbar
2	CStatusbarEX()	method	void	Hàm khởi tạo cửa sổ Statusbar không đối số
3	~CStatusbarEX()	method	void	Hàm hủy cửa sổ Statusbar không đối số
4	CreateMainFrame()	method	HWND	Hàm tạo cửa sổ Statusbar có đối số

#	Tên	Tham số	Kiểu trả về	Mô tả
1	Resize()	HWND hwnd	void	tính toán, và thay đổi kích thước của các cửa sổ cho phù hợp khi cửa sổ chính thay đổi.
2	ShowMea nEV()	HWND hwndRichedit HWND hWnd ifstream &filein	void	Hàm lấy dữ liệu trong file và hiển thị lên Richedit
3	Loadadata() loadadata2()	HWND hwndListbox ifstream &Filein	void	Lấy và hiển thị toàn bộ dữ liệu (từ điển) lên Listbox
4	Convertost ring()	TCHAR tcArr[]	string	Chuyển đổi kiểu dữ liệu TCHAR* sang string
5	*Convertot char()	string strA	TCHAR *	Chuyển đổi kiểu dữ liệu string sang TCHAR*
6	CreateWri teFileUTF _8()	string s TCHAR *swork	void	Tạo 1 file kiểu UTF_8 (file name: s), và ghi dữ liệu swork vào file đó.
7	*ReadRic hEdit()	HWND hwndRichedit	TCHAR *	Đọc dữ liệu hiện có ở Richedit
8	AddNew Word()	không	void	Thêm 1 từ
9	UpdateSta tus()	không	void	Cập nhật lại trạng thái và số từ có trong Listbox
10	AutoSoft()	Không	void	Sắp xếp theo bảng chữ cái của danh sách.

BẢNG 3. 6: mô tả các class diagram và functions

3.2.2 Lập trình hiển thị giao diện

Tạo giao diện của ứng dụng:

- Tạo cửa sổ *Windows*
- Tạo *Menu bar, Status bar*
- Tạo *Toolbar*
- Tạo *Combobox*
- Tạo *Button*
- Tạo *Listbox*
- Tạo *Richedit*

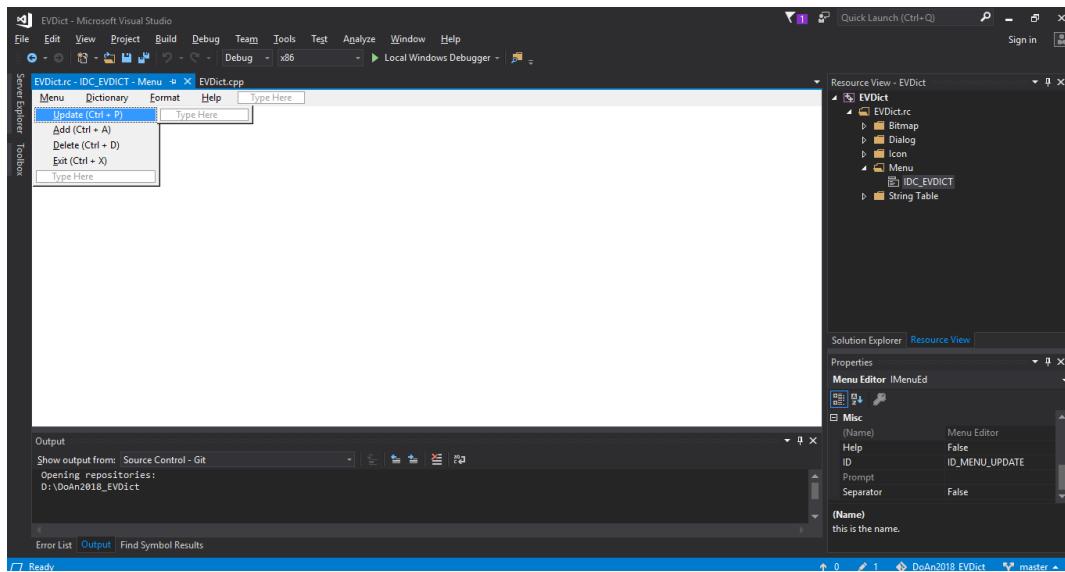
Khi tạo *project* trên môi trường *Windows Desktop Application*, thì sẽ có một cửa sổ *windows* mặc định được tạo ra.

Tạo *menu bar*: sử dụng *tool* trên môi trường lập trình để dễ dàng tạo *menu bar*:

Resource View → *EVDict* → *EVDict.rc* → *Menu* → *IDC_EVDICT*

Khi tạo *menu bar*: cần đặt *ID* cho từng thành phần của *menu* ở mục:

Properties, để sau sử lý thao tác chọn lên thành phần đó của *menu*.



HÌNH 3. 21: Tạo menu bar

Tương tự như vậy, ta lần lượt tạo và thêm các thành phần của *menu bar*.

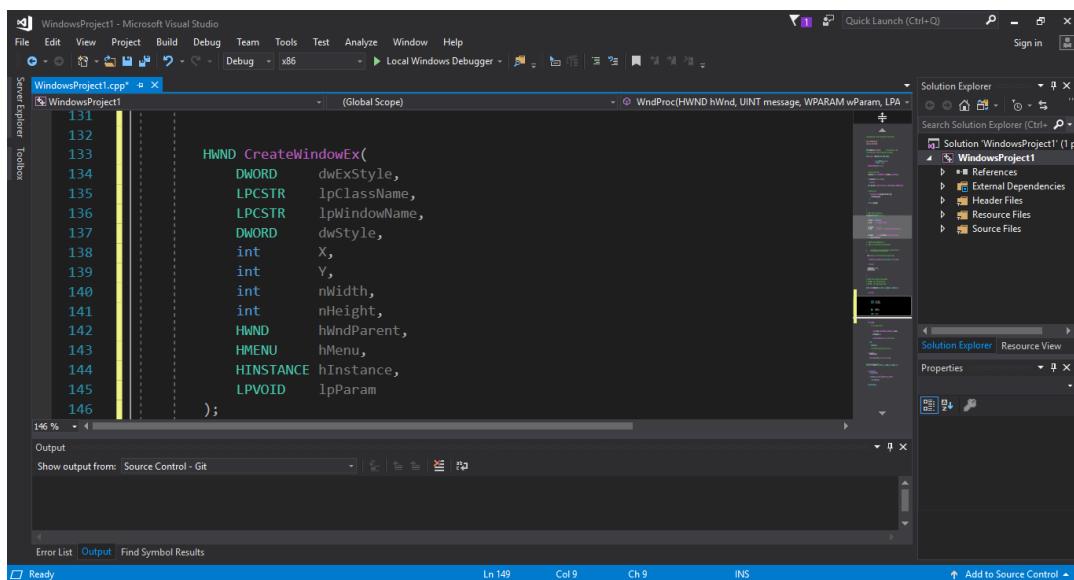
Để tạo *Toolbar, Combobox, Button, Listbox, Richedit*, có nhiều cách để tạo cửa sổ, trong tài liệu này, tôi sử dụng hàm *CreateWindowEx()* để tạo cửa sổ.

Với các tham số truyền vào:

- *dwExStyle*: kiểu sửa sổ mở rộng được tạo.
 - *lpClassName* (loại cửa sổ):
 - **TOOLBARCLASSNAME**: *Toolbar*
 - **_T("Combobox")**: *Combobox*
 - **_T("listbox")**: *Listbox*
 - **MSFTEdit_CLASS**: *Richedit*
 - **STATUSCLASSNAME**: *Status bar*
 - **_T("button")**: *Button*
 - *lpWindowName*: tên cửa sổ
 - *dwStyle*: kiểu cửa sổ đang được tạo
 - *x*: vị trí nằm ngang ban đầu của cửa sổ. Đối với một cửa sổ chồng chéo hoặc cửa sổ bật lên, thông số *x* là toạ độ *x* ban đầu của góc trên bên trái của cửa sổ, trong tọa độ màn hình. Đối với một cửa sổ con, *x* là tọa độ *x* của góc trên bên trái của cửa sổ tương ứng với góc trên bên trái của khu vực khách hàng của cửa sổ mẹ.
 - *y*: ví trí thẳng đứng ban đầu của cửa sổ. Đối với một cửa sổ chồng chéo hoặc cửa sổ bật lên, tham số *y* là toạ độ *y* ban đầu của góc trên bên trái của cửa sổ, trong tọa độ màn hình. Đối với cửa sổ con, *y* là toạ độ *y* ban đầu của góc trên bên trái của cửa sổ con tương ứng với góc trên bên trái của khu vực khách sổ của cửa sổ mẹ.
 - *nWidth*: chiều rộng, tính bằng đơn vị thiết bị, của cửa sổ.
 - *nHeight*: chiều cao, tính bằng đơn vị thiết bị, của cửa sổ.
 - *hWndParent*: một xử lý cho cửa sổ cha hoặc cửa sổ chủ sở hữu của cửa sổ đang được tạo. Để tạo cửa sổ con hoặc cửa sổ được sở hữu, hãy cung cấp một cửa sổ hợp lệ. Tham số này là tùy chọn cho các cửa sổ bật lên.
 - *hMenu*: một xử lý cho một trình đơn, hoặc chỉ định một định danh cửa sổ con, tùy thuộc vào kiểu cửa sổ. Nói cách khác, đây cũng chính là *ID* của cửa sổ con.

của cửa sổ, sau sẽ được xử lý trong tin nhắn *WM_COMMAND* của *WinProc*.

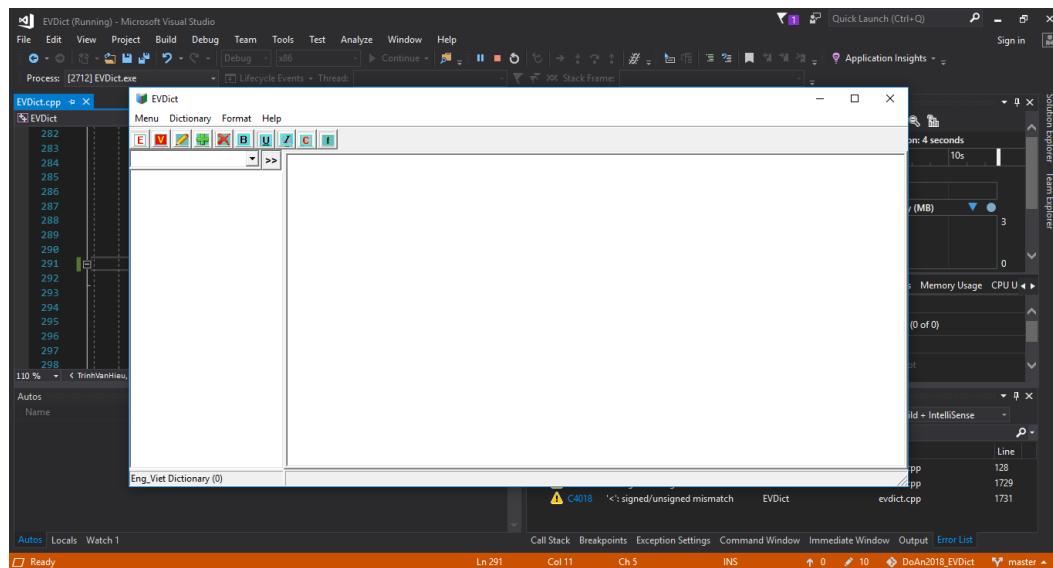
- *hInstance*: một xử lý đối với cá thể của module được liên kết với cửa sổ.
- *lParam*: con trỏ tới một giá trị được chuyển đến cửa sổ thông qua cấu trúc *CREATESTRUCT*. Thông báo này được gửi đến cửa sổ được tạo bởi hàm này trước khi nó trả về.



HÌNH 3. 22: *Hàm tạo cửa sổ*

Như vậy, để tạo giao diện hiển thị, là tổng hợp của các cửa sổ, do đó ta phải tùy chỉnh tọa độ *x*, *y* trong trong hàm tạo để được giao diện phù hợp nhất.

Một điều chú ý khi tạo Toolbar, để chèn hình ảnh (*bitmap*) vào *toolbar*, ta phải tạo một danh sách lưu trữ các hình ảnh, có *pixel* phù hợp.



HÌNH 3. 23: Giao diện hiển thị sau khi tạo

3.2.3 Tiến hành lập trình

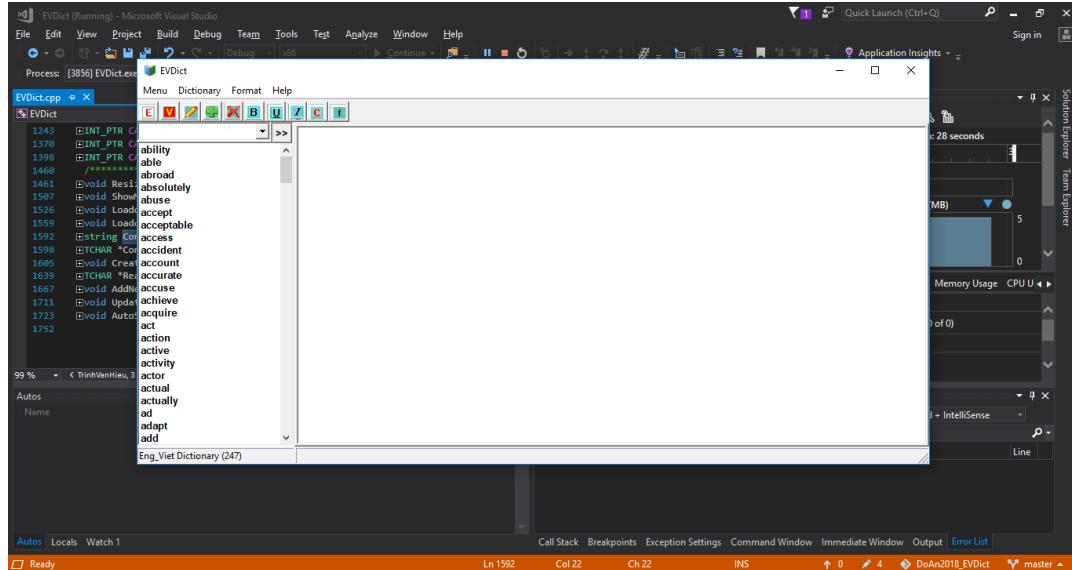
3.2.3.1 Lập trình hiển thị danh sách từ điển khi mở ứng dụng.

Về ý tưởng cơ bản, để hiển thị được danh sách các từ điển lên *Listbox*, ta cần đọc một tệp (*file*) có lưu trữ toàn bộ danh sách từ điển, và hiển thị nó lên *Listbox*. Có nhiều phương pháp đọc ghi file dữ liệu, để đơn giản nhất, ta sử dụng kiến thức đọc ghi file cơ bản của C++, đó là sử dụng các hàm, tính năng trong thư viện *fstream.h*.

Một điểm chú ý, khi ta sử dụng phương pháp trên để đọc *file*, thì dữ liệu thu được thuộc kiểu *string*, hoặc *char**. Nhưng để hiển thị dữ liệu lên *Listbox*, thì dữ liệu cần truyền vào để hiện thị là kiểu *TCHAR**. Do vậy, sẽ phải xây dựng các hàm chuyển đổi từ kiểu dữ liệu khi đọc file sang kiểu dữ liệu có thể hiển thị trên *Listbox* (*string* → *TCHAR**, *TCHAR** → *string*). Và lưu trữ trong một *vector*. Lựa chọn *vector* để lưu trữ, và quản lý danh sách từ điển là vì: *vector* là sự kết hợp các ưu điểm của mảng, và con trỏ. Vector không cần phải cố định số lượng phần tử ban đầu, có thể thêm kích thước sau khi sử dụng, có khả năng truy suất dữ liệu nhanh.

Để hiện thị lên *Status bar*, ta sử dụng hàm đếm số phần tử có trong *Listbox*, và trạng thái chế độ đang tra cứu, rồi hiển thị lên *Statusbar*.

Sau lập trình hiển thị được danh sách từ điển, giao diện hiển thị sau khi mở ứng dụng lên sẽ có hình như sau:



HÌNH 3. 24 giao diện hiển thị chính

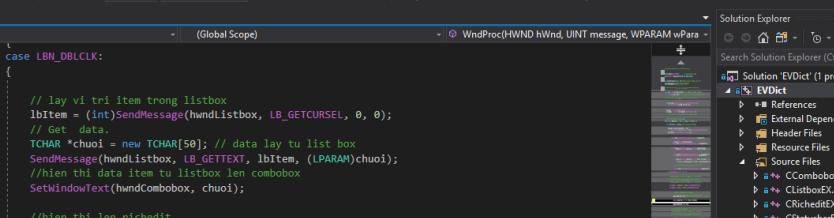
3.2.3.2 Lập trình tính năng tra cứu từ điển.

Theo như (Hình 3.5: sơ đồ luồng thực thi tìm kiếm), đó chính là luồng code chạy trong quá trình tìm kiếm từ điển. Đối với lập trình *Windows Destop Application*, mọi tác động, hành động, thao tác lên bất kì cửa sổ, chúng đều là một *message*. Do vậy, để thực thi một thao tác, ta sẽ thực thi code trên message tương ứng của hàm *WinProc()*. Cụ thể là trong *message WM_COMMAND*.

Về ý tưởng hiển thị nghĩa của từ lên *Richedit*, tương tự như hiển thị danh sách từ điển lên *Listbox*, ta cũng sẽ có một *file* chứa nghĩa của từ đó. Có nhiều cách, nhưng để đơn giản, và thuận tiện cho tính năng *Update* nghĩa của từ, thì đối với mỗi từ sẽ là một *file*, và tên của *file* chính là từ đó.

Tìm kiếm từ bằng cách *double click* lên từ đó trên *Listbox*, hoặc gõ từ vào *Combobox* sau đó bấm *Button* tìm kiếm, bằng cách sử các hàm mà *Listbox*, và *Combobox* hỗ trợ, ta có thể xác định được ví trí, và nội dung được chọn. Từ đó mở *file* có tên *file* là từ đó, rồi hiển thị lên *Richedit*, nếu không tìm thấy sẽ có thông báo tới người sử dụng.

Cụ thể, đối với *Listbox*, khi *double click* lên một từ trong *Listbox*, sử dụng *message LB_GETCURSEL*, *LB_GETTEXT*, để lấy vị trí và nội dung từ được chọn.



```
EVDict - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Server Explorer Tools
EVDict.cpp * x
EVDict (Global Scope)
109 % TrimVnHienti, 12 hours ago | author, 16 changes
File List Output Find Symbol Results
Ln 629 Col 22 Ch 7 INS 0 2 DoAn2018.EVDict master
case LBN_DBLCLK:
{
    // lay vi tri item trong listbox
    lItem = (int)SendMessage(hwndListbox, LB_GETCURSEL, 0, 0);
    // Get data.
    TCHAR *chuoi = new TCHAR[50]; // data lay tu list box
    SendMessage(hwndListbox, LB_GETTEXT, lItem, (LPARAM)chuoi);
    //hien thi data tu listbox len combobox
    SetWindowText(hwndCombobox, chuoi);

    //hien thi len richedit
    string a;
    if (flag == 1)
    {
        a = "Data/Eng_Viet/Data/";
    }
    else
    {
        a = "Data/Viet_Eng/Data/";
    }
    string b = Converttostring(chuoi);
    string c = ".txt";
    string buff = a + b + c;

    ifstream Filein(buff, ios_base::in);
    ShowMeanV(hwndRichedit, hwnd, Filein);
    Filein.close();
}

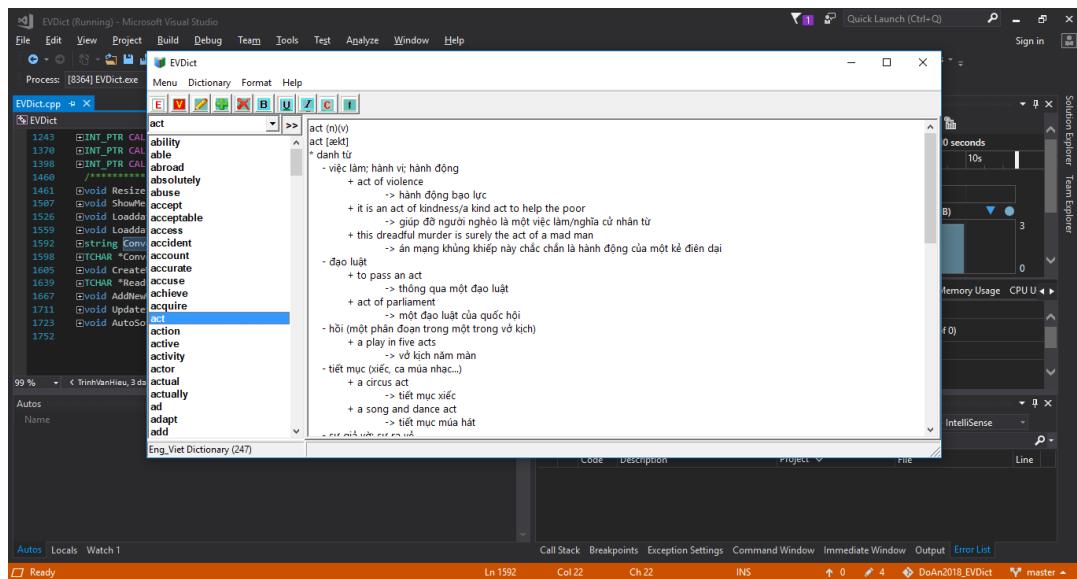
```

HÌNH 3. 25: xác định từ trong Listbox và mở file lấy nghĩa

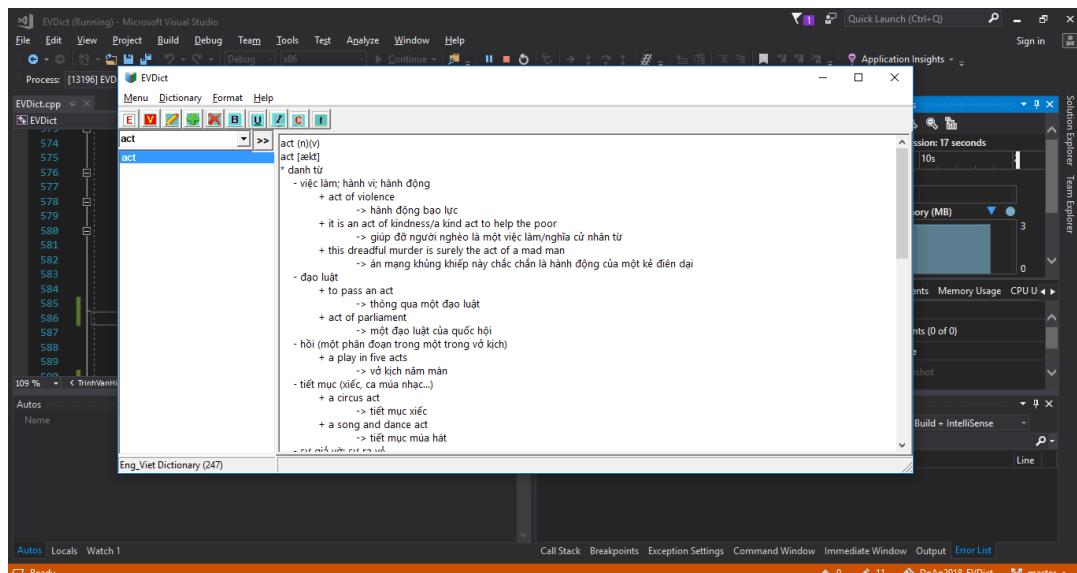
Đối với *Combobox*, cũng làm tương tự như *Listbox*, sử dụng hàm *GetWindowText()* để lấy dữ liệu đã nhập vào ô *Combobox*. Dữ liệu đó được so sánh với danh sách từ điển, nếu trùng một trong số các từ thì sẽ mở file và hiển thị nghĩa lên *Richedit*. Sau khi tìm thấy một từ, thì sẽ lưu từ đó trong lịch sử của *Combobox*.

Để thực hiện điều này, ta sẽ sử dụng thêm một *vector* để lưu trữ lịch sử tìm thấy, và kiểm tra xem từ các từ sau tìm kiếm, nếu đã được tìm kiếm trước đó thì sẽ không lưu lại lịch sử.

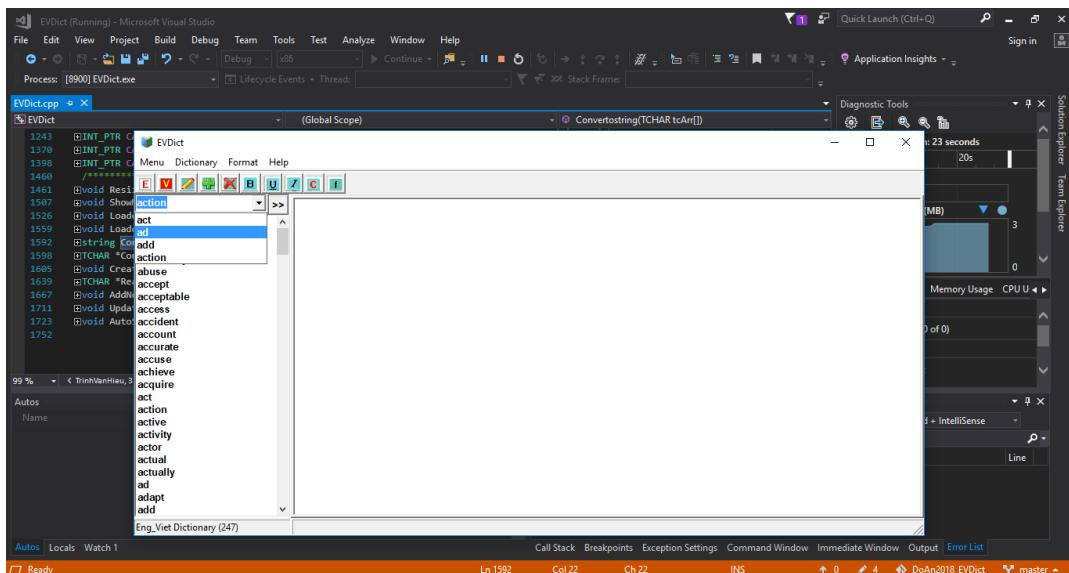
Mỗi liên quan giao diện giữa *Listbox* và *Combobox*, khi một từ được chọn ở *Listbox* thì từ đó cũng sẽ hiển thị lên ô của *Combobox*. Khi gõ và tìm được từ trên *Combobox*, thì từ đó sẽ được chọn trên *Listbox*.



HÌNH 3. 26: tìm kiếm từ trên Listbox



HÌNH 3. 27: tìm kiếm từ trên Combobox



HÌNH 3. 28: lịch sử tìm kiếm trên Combobox

3.2.3.3 Lập trình tính năng xóa từ điển

Để xóa từ với hai chế độ, là xóa tạm thời trong ứng dụng, và xóa vĩnh viễn khỏi ứng dụng. Để làm được điều này, đối với mỗi từ được xác định trong danh sách thì đều có hai thành phần: là nội dung từ, và *flag* (cờ). Nếu từ bị xóa tạm thời thì cờ sẽ được đánh dấu, để sau khi hiển thị lại danh sách, nếu từ nào bị đánh dấu thì sẽ không cho hiển thị lên danh sách.

Để thực hiện xóa từ, ta xác định từ cần xóa, bằng cách tìm kiếm từ điển, sau đó xóa khỏi danh sách từ điển (với vector thì đơn giản chỉ cần dùng hàm *erase()*).

Đối với trường hợp xóa vĩnh viễn từ ra khỏi ứng dụng, ta sẽ thực hiện xóa từ đó trong file lưu trữ. Xác định từ cần xóa, ghi đè lên file dữ liệu từ điển.

Sau đó sẽ có thông báo tới người dùng và hiển thị lại danh sách mới lên *Listbox*.

Toàn bộ lựa chọn của người dùng sẽ được lựa chọn thông qua một *dialog* lựa chọn thao tác xóa từ.

3.2.3.4 Lập trình tính năng cập nhật nghĩa của từ.

Tương tự như tính năng xóa từ, trước hết để cập nhật nghĩa cho từ điển. Xác định từ cần cập nhật nghĩa, sau đó trực tiếp cập nhật nội dung nghĩa của từ đó trên màn hình hiển thị nghĩa (*Richedit*).

Để làm được điều này, ta cũng sử dụng phương pháp ghi đè dữ liệu lên *file* nghĩa của từ đó.

3.2.3.5 Lập trình tính năng thêm từ.

Thêm từ là tính năng phức tạp nhất trong các tính năng.

Khi thêm một từ, sẽ xảy ra nhiều trường hợp: từ đó đã tồn tại trong danh sách, từ đó đã bị xóa tạm thời (không phải xóa vĩnh viễn, khi hiển thị lại danh sách, thì từ đó vẫn hiển thị), hoặc từ đó chưa tồn tại trong danh sách.

Đối với từ chưa có trong danh sách, ta chỉ việc thêm từ đó vào trong danh sách quản lý (với *vector* thì dùng hàm *pushback()*). Tiếp đến, mở *file* danh sách từ điển, và *file* nghĩa của từ để ghi từ đó vào *file* dữ liệu của ứng dụng.

Đối với từ đã tồn tại trong danh sách từ điển, thì sẽ không thêm, và gửi thông báo và *dialog* cho người dùng lựa chọn có *update* nghĩa cho từ đó không.

Đối với từ vừa bị xóa tạm khỏi ứng dụng, thì bật lại cờ cho từ đó, hiển thị lại và thông báo cho người dùng. Hay nói cách khác, bỏ xóa tạm thời đối với từ đó.

3.2.3.6 Lập trình cài đặt các *hot key* (phím tắt)

Tương tự như cách lập trình trên, chúng ta đều viết *code* thực thi chương trình nằm trong hàm *WndProc()*.

Đối với lập trình tạo giao diện, hiển thị danh sách từ điển lên giao diện ban đầu, thì sẽ ở *message WM_CREATE*.

Đối với các lập trình tra cứu từ, cập nhật nghĩa, xóa từ hay thêm từ, thì sẽ xử lý các *message* (chính là *ID* của từng cửa sổ, thao tác) trong *WM_COMMAND*.

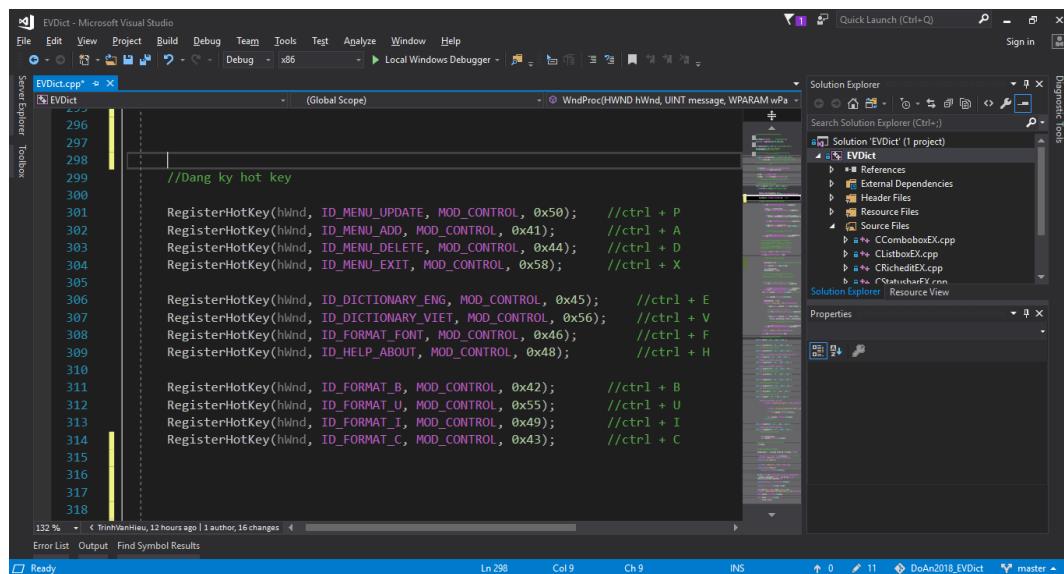
Các *message* này sẽ nhận được, khi người dùng tác động, thao tác lên chúng.

Bằng cách lấy dữ liệu ở tham số *wParam* (các bit thấp), sử dụng hàm *LOWORD(wParam)* để trả về *ID* cần xử lý.

Nhận thấy, các thao tác của người dùng lên ứng dụng, sẽ là *click* chuột chọn cửa sổ, hay gõ từ cũng hơi mất thời gian. Như vậy, để sử dụng một cách nhanh chóng, các tính năng hay dùng, thì ta sẽ cài đặt các *hot key* (tổ hợp phím tắt) để thao tác nhanh hơn. Các thao tác có thể được chọn thông qua tổ hợp phím tắt trên bàn phím.

Để làm được điều này, sử dụng hàm đăng ký phím tắt *RegisterHotKey()* trong *WM_CREATE* của *WndProc()*.

Cụ thể cách dùng, thao tác với các tham số để đăng ký các *hot key* thì tham khảo ở phần phụ lục.



```

296
297
298
299
300
301 RegisterHotKey(hWnd, ID_MENU_UPDATE, MOD_CONTROL, 0x50); //ctrl + P
302 RegisterHotKey(hWnd, ID_MENU_ADD, MOD_CONTROL, 0x41); //ctrl + A
303 RegisterHotKey(hWnd, ID_MENU_DELETE, MOD_CONTROL, 0x44); //ctrl + D
304 RegisterHotKey(hWnd, ID_MENU_EXIT, MOD_CONTROL, 0x58); //ctrl + X
305
306 RegisterHotKey(hWnd, ID_DICTIONARY_ENG, MOD_CONTROL, 0x45); //ctrl + E
307 RegisterHotKey(hWnd, ID_DICTIONARY_VIET, MOD_CONTROL, 0x56); //ctrl + V
308 RegisterHotKey(hWnd, ID_FORMAT_FONT, MOD_CONTROL, 0x46); //ctrl + F
309 RegisterHotKey(hWnd, ID_HELP_ABOUT, MOD_CONTROL, 0x48); //ctrl + H
310
311 RegisterHotKey(hWnd, ID_FORMAT_B, MOD_CONTROL, 0x42); //ctrl + B
312 RegisterHotKey(hWnd, ID_FORMAT_U, MOD_CONTROL, 0x55); //ctrl + U
313 RegisterHotKey(hWnd, ID_FORMAT_I, MOD_CONTROL, 0x49); //ctrl + I
314 RegisterHotKey(hWnd, ID_FORMAT_C, MOD_CONTROL, 0x43); //ctrl + C
315
316
317
318

```

HÌNH 3. 29: đăng ký các hot key

Để hot key có thể hoạt động thay thế cho một tính năng, thao tác, thì ID của hot key phải trùng với tính năng, và thao tác đó.

Viết code thực thi tính năng của hot key, tương tự như ở *WM_COMMAND*, *hot key* cũng có *WM_HOTKEY*, và cũng lấy dữ liệu trả về từ các bit thấp của tham số *wParam* để xác định hot key nào đang được ấn.

3.3 KIỂM TRA, ĐÁNH GIÁ VÀ ĐÓNG GÓI ỨNG DỤNG

3.1.1 Kiểm tra, đánh giá ứng dụng

Ứng dụng chạy ổn định, phản hồi nhanh, dung lượng bộ nhớ chiếm ít (khoảng 3MB). Giao diện hiển thị dễ nhìn, dễ thao tác.

Đánh giá: về cơ bản, ứng dụng đáp ứng được yêu cầu thiết kế, và các tính năng chính của ứng dụng.

3.1.2 Đóng gói ứng dụng

Sử dụng tính năng đóng gói ứng dụng trên Visual studio để tạo thành một ứng dụng, có thể cài đặt, và chạy trên các môi trường window.

KẾT LUẬN

Ứng dụng đáp ứng được các yêu cầu cơ bản của một chương trình tra cứu từ điển Anh – Việt, Việt – Anh theo chuẩn Dict. Như người dùng có thể lựa chọn chế độ sử dụng, tra cứu từ điển, thêm từ, xóa và cập nhật lại nghĩa của từ điển. Bên cạnh đó, giao diện trực quan, dễ sử dụng đối với người dùng.

Tuy nhiên, ứng dụng vẫn còn nhiều hạn chế như chức năng *Format*, *font* chữ, và màu chữ còn chưa được thêm vào.

Do kiến thức và trình độ còn hạn chế, nên trong báo cáo đồ án vẫn còn nhiều khiếm khuyết, em rất mong được sự chỉ bảo của các thầy, cô để đồ án em được hoàn thiện hơn. Em xin chân thành cảm ơn các thầy, cô và các bạn trong lớp đã tạo điều kiện để em hoàn thành đúng quy định đồ án của mình. Chi tiết về code đồ án của em xin đính kèm ở phần phụ lục.

PHỤ LỤC

[1] Project:

https://github.com/TrinhVanHieu/DoAn2018_EVDict

[2] Các bài viết tham khảo về ngôn ngữ lập trình C, C++, Win32.

https://vietjack.com/lap_trinh_c/

<https://vietjack.com/cplusplus/index.jsp>

<http://vncoding.net/lap-trinh-win32-api/>

<https://www.learncpp.com/>

[3] Các cách tạo cửa sổ, đăng ký hot key:

<https://docs.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-createwindowexa>

<https://docs.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-registerhotkey>

DANH MỤC TÀI LIỆU THAM KHẢO

Tiếng việt:

- [1] Kỹ thuật lập trình C cơ bản và nâng cao, GS. Phạm Văn Át, Giao thông vận tải, 2015, 430 trang.
 - [2] Ngôn ngữ lập trình C++ từ cơ bản đến hướng đối tượng, TS. Dương Tử Cường, Khoa học và kỹ thuật, 2005, 505 trang.
 - [3] Lập trình WIN32 API qua ví dụ minh họa, Vũ Hồng Việt, Vncoding.net, 2014, 51 trang.

Tiếng anh:

- [1] C and C++ Compiling, Milan Stevanovic, Apress, 2004, 326 pages.
 - [2] The C++ Programming Language, Bjarne Stroustrup, Addison – Wesley, 1997, 1022 pages.