Mục lục

Lời nói đầu

Super Power Dict (SPDict) là một từ điển mở về mã nguồn (không dùng bất cứ biện pháp nào bảo vệ mã nguồn) , dữ liệu cũng như thuật toán làm từ điển với tinh thần chia sẻ kiến thức cũng như kinh nghiệm của bản thân cho các bạn có hứng thú với vấn đề này , tuy nhiên từ phiên bản java (6.0) này tớ sẽ không cung cấp source code như nhiều phần mềm nguồn mở hiện nay để down về biên dịch mà các bạn muốn có source code thì dùng các trình dịch ngược file java ví dụ như : DJ Java Decompiler.

Nguyên nhân của việc làm này là tớ muốn truyền đạt kiến thức cho những người thực sự muốn tìm hiểu , có một kỹ năng code nhất định và có khả năng phát triển để đóng góp thêm cộng đồng nguồn mở chứ không phải dành cho những người chỉ biết ăn sẵn , bốc code về sửa tên rồi đem đi khoe cái này là của mình , thậm chí còn chẳng biết code java (T_T) , hoặc biết code nhưng được ăn sẵn nên đâm ra lười , dịch được , chạy được là cho vào ứng dụng của mình mà chẳng nghiên cứu được gì về từ điển hết .

Tài liệu này được trình bày là toàn bộ những gì mình muốn chia sẻ kiến thức , không hỗ trợ thêm riêng cho ai , những thứ không được trình bày trong tài liệu thì một là không muốn truyền đạt , hai là quá dễ , quá cơ bản không đáng để truyền đạt , vì vậy các bạn gửi yêu cầu giúp đỡ riêng thì tuyên bố trước là mình không làm đâu .

Thông cảm , mình hơi rắn trong việc này vì người cần hỗ trợ thì sướng rồi , có mấy ai hiểu được nỗi lòng của người suppost đâu (^_^), 1 số trường hợp xin xong rồi còn phủi tay , chê này chê nọ , phủ nhận những thứ người khác phải mất nhiều công sức để làm ra , có người thì chả biết gì thì biết hỗ trợ cái gì đây , chả lẽ làm hộ ….. ngoài ra tớ còn mất thời gian để giải đáp vì ai cũng chỉ muốn hỗ trợ riêng , không chịu tìm kiếm , add nick , gửi mail spam lung tung , tớ từng phải bỏ 1 nick yahoo vì những lý do này .

Giới thiệu

Giới thiệu qua chút về bản thân:
Họ tên : Bùi Đức Tiến
Năm sinh 1989
web :
http://superpowerdict.googlepages.com/
http://tienlbhoc.vnbb.com/

Làm từ điển nếu chỉ với các chức năng cơ bản thì không phải là cái gì phức tạp lắm , nhưng đi sâu vào để hoàn thiện đầy đủ thì lại rất khó khăn . Đề tài này được nhiều sinh viên CNTT chọn làm đề tài làm bài tập lớn nhưng số lượng phần mềm mã nguồn mở hay miễn phí có chất lượng thật sự không nhiều. Mình muốn làm một cái , giá mà giống unikey dù ít tính năng hơn nhưng ổn định thay thế được vietkey thì tốt.

Hiện nay có rất nhiều từ điển online , không phải cài đặt, cập nhật từ thường xuyên , miễn phí , nhưng nó cũng có những yếu điểm không thể thay thế được từ điển cài đặt vì vậy việc làm từ điển này vẫn rất cần thiết vì:

+Không phải ai cũng lên net để tra và không phải ai cũng có net , chưa kể nhiều người xót tiền net nữa

+Đôi khi người dùng chỉ muốn tự tạo riêng cho mình để ghi chép vấn đề gì đó hay chuyên ngành gì đó cho riêng mình hay chia sẻ nội bộ . Từ điển online hiện giờ không thể làm được điều đó

+Vì là web nên ví dụ các tính năng kiểu như click and see , load đầy đủ danh sách ... không thể có

+Web có thể dừng hoạt động bất cứ lúc nào

+Dữ liệu thêm vào trang web sẽ không lấy lại được, như vậy , phải gắn bó với nó mãi mãi , giống như google không thể chat với yahoo , khi người dùng không thích dùng web này , thì khi sang web khác, có thể sẽ phải nhập lại các từ trước kia .

Đó là lý do người dùng ít nhập từ vào từ điển online mà cần phải có một từ điển dữ liệu mở như SPDict

Một số từ điển cùng loại

Từ điển offline là các từ điển chạy trên máy mà không cần kết nối net đó:

Hiện tại nổi tiếng nhất trong giới mã nguồn mở là Stardict , một phần mềm có khả năng tra từ khá nhanh , gọn nhẹ, bắt từ trong ứng dụng khác (click and see) khá tốt ,định dạng của nó là một biến thể đã nâng cấp của chuẩn Dict , một chuẩn mà trong thời điểm hiện tại đa số các phần mềm miễn phí , nguồn mở đều dùng để làm từ điển , nhưng nó cũng có những nhược điểm nhất định chưa khắc phục được của chuẩn Dict đó là dữ liệu từ điển phải ghi vào trong nhiều file dữ liệu khác nhau , như trong stardict là 4 file. Ngoài ra khả năng thêm xoá không có cũng là một hạn chế rất lớn của chuẩn từ điển này và các biến thể của nó .

Ngoài các soft mã nguồn mở , phải kể đến một bộ phận không nhỏ là soft từ điển miễn phí với đại diện mà mình cho là tốt nhất là lingoes , với các tính năng cũng khá giống stardict nhưng có ưu điểm là phần danh sách từ của nó xuyên suốt từ đầu đến cuối , có phát âm công nghệ text to speech của microsoft , còn stardict chỉ có 30 từ trong danh sách. Nhưng nó cũng có nhược điểm là tác giả của nó không cung cấp bộ công cụ convert dữ liệu từ điển, phải gửi dữ liệu đã biên soạn bằng text cho tác giả để tác giả convert thành ra hiện tại bộ dữ liệu từ điển của nó không nhiều ngôn ngữ và hiện tại stardict vẫn phổ biến hơn.

Đối với các phần mềm miễn phí trong nước thì có các phần mềm :Multidictionary (phổ biến nhất) , powerclick và jtranslator mới xuất hiện.

Tất cả các từ điển trên đều không có khả năng thêm xoá từ và tạo từ điển .
Hi vọng từ điển này ra đời sẽ khoả lấp được chỗ trống đó . Đồng thời với cơ chế dữ liệu mở và mã nguồn mở sẽ thúc đẩy được sự xuất hiện của những từ điển miễn phí , nguồn mở tăng thêm về chất lượng , tính năng phục vụ cho cộng đồng . Trong thời buổi hội nhập này , việc học ngoại ngữ rất quan trọng và công việc này của mình chắc cũng không phải là vô nghĩa . Tuy nói vậy thôi chứ mình tiếng Anh dốt lắm (^_^) .

Giới thiệu luôn cho mọi người một số từ điển thương mại phổ biến hiện nay:
+Lạc việt mtd : có thêm xoá từ , tra từ click and see , chạy ổn định , dữ liệu từ điển tương đối đầy đủ .

+Just click n see: chỉ có tra từ click and see(tốt hơn lạc việt) , rất ít tính năng.

+Evtran 2.0 : có thêm xoá, nó là phần mềm dịch thì đúng hơn nhưng bản 3.0 không có khả năng thêm xoá, chức năng

click and see lên vista thì liệt. các bạn có thể vào trang vdict.com để dịch trực tuyến miễn phí

+English Study 4.0: đây là phần mềm ngữ pháp tiếng anh + từ điển + luyện nghe

+babylon: không có khả năng thêm xoá nhưng khả năng click and see thuộc loại tốt nhất hiện nay cùng với khả năng tìm kiếm từ gần đúng hoàn hảo, nó rất mạnh .

+prodict, và javidict thuộc cùng một hãng, đặc điểm của loại này là dữ liệu lớn nhất hiện nay gồm nhiều chuyên ngành, nhưng không thêm xoá (javidict thì mình không biết), tra từ click and see ngang lạc việt , hiện nay trang http://tratu.baamboo.com/ đã mua bản quyền dữ liệu và các bạn có thể tra từ trực tuyến trên đó miễn phí.

Đây là bảng so sánh:

	Dữ liệu	Click and see	Tìm kiếm	Phát âm	Cập nhật dữ liệu	File data	Load danh sách
Stardict	Nhiều ngôn ngữ	ổn	Tra đa từ điển, Có truy vấn mờ tạm được, tra wildcard (tra với * và ?)	Giọng thật dung lượng lớn , ít mỗi tiếng anh	Không thêm xoá, có tool tạo từ điển	4 file	30 từ , danh sách chung cho các từ điển
lingoes	ít	ổn	Tra đa từ điển	Text to speech (TTS 4 và 5)	Không, phải gửi data cho tác giả để tạo	1 file cài đặt	Load đầy đủ , nhanh
Multi dictionary	ít	Tàm tạm	Tra tối đa 3 từ điển,tìm kiếm thông minh kém	TTS4	Hạn chế rất nhiều	4 file	Đầy đủ nhưng chậm
powerclick	Rất ít	Tàm tạm	Có 1 từ điển , không tìm thông minh	TTS4 , mỗi tiếng anh	không	3 file	Đầy đủ, nhanh
jtranslator	Bình thường	Tra qua clipboard	1 từ điển , tìm kiếm thông minh tàm tạm	Free java TTS	không	3 file	Đầy đủ , nhanh
mtd	Nhiều, chất lượng tốt	Trung bình	1 từ điển, tra chéo , cho dùng wildcard	TTS	Có , không có tool convert	1 file	Đầy đủ, nhanh
Just click n see	Rất ít	khá	kém	không	không	Không rõ	Đầy đủ, nhanh
Evtran 2.0	Bình thường	không	1 từ điển , tra wildcard	không	có	Nhiều file	Đầy đủ, nhanh
Evtran 3.0	Bình thường	Trung bình	2 từ điển	không	không	Nhiều file	Không có
EStudy	Hơi ít	Trung bình	1 từ điển, tìm thông	Giọng thật ,	Có , không	Nhiều file	Đầy đủ , nhanh

			minh kém	chỉ tiếng Anh	tool convert		
babylon7	Nhiều ngôn ngữ	Rất tốt	Tra đa từ điển, tìm thông minh rất tốt	TTS 4 + 5	Không , có tool convert	1 file cài đặt	Nhiều từ điển, load ít từ , danh sách chung
Prodic	Dữ liệu lớn nhất	Bình thường	Bình thường , tìm cả phần từ và nghĩa nhưng chậm	TTS	Không, chỉ cho tìm online	1 file	Đầy đủ , nhanh
SPDict	Nhiều ngôn ngữ	Tra qua clipboard	Tra đa từ điển, tìm kiếm thông minh với wildcard , regular expression tìm từ gần đúng .	Java text to speech	Có , có tool convert	1 file	2 loại : Đầy đủ với spdict Chung với spdict small

Ngoài các so sánh trên spdict còn có 1 ưu điểm là chạy đa nền tảng , chỉ cần có java runtime thì dù win hay linux đều chạy được hết

Những kiến thức cần trang bị

Dữ liệu từ điển

Bắt đầu nhé, để có thể làm một từ điển, ngoài việc có một kỹ năng lập trình, một thuật toán tìm kiếm nhanh, một cấu trúc dữ liệu .... thì cái cần nhất là CSDL từ điển, nhập một CSDL từ đầu thì đúng là mệt, chưa kể nó vừa ít về số lượng, kém về nội dung, lại có thể sai về ngữ nghĩa (con người mà, sai là chuyện thường), rất may cho chúng ta, có một nguồn cung cấp từ điển rất lớn trên web của người việt đó là ở trang www.tudientiengviet.net, số lượng từ điển ở đây rất phong phú , đa ngôn ngữ và thừa để bạn có thể bắt tay vào làm soft từ điển.
Để lấy CSDL từ điển bạn vào tải dữ liệu stardict (một từ điển nguồn mở khá phổ biến nhất là trong linux)

http://www.tudientiengviet.net/data.html

Để dùng dữ liệu stardict bạn hãy dùng công cụ stardict-editor

nó là một công cụ convert file stardict sang định dạng dict.tab và ngược lại.

file dict.tab sau khi convert từ stardict sẽ là file để lấy dữ liệu cho từ điển của chúng ta bởi vì định dạng của nó cực kỳ đơn giản và nó còn có một số tính năng bổ trợ từ điển rất tốt
Đây là trích nguyên văn của định dạng này:

> :Here is a example dict.tab file
> ==========
> a 1\n2\n3
> b 4\\5\n6
> c 789
> ==========
> It means: write the search word first, then a Tab character, and the definition. If the definition contains

. new line, just write \n, if contains \ character, just write

Bài viết đầu tiên mình giới thiệu qua thế, để mọi người có thể định hướng, nhưng mình nói trước, plain text rất dễ hiểu nhưng không bao giờ nên dùng nó làm từ điển vì tốc độ sẽ rất chậm, mình đã thử, nếu chỉ tra một từ điển nhỏ khoảng 30.000 từ thì còn được, tra vài từ điển lớn cùng một lúc (như babylon 17 ngôn ngữ) thì nguy, đấy là chưa kể người dùng của bạn có máy cấu hình thấp.

Lưu ý:

+File stardict có 3 hoặc 4 file , để decompile bạn phải chọn file có đuôi ifo

+đôi khi stardict convert lỗi với nội dung như sau:

> Building...
> File not exist: D:\YViet\star_yviet.dict
> Please rename somedict.dict.dz to somedict.dict.gz and use SevenZip to uncompress the somedict.dict.gz file, then you can get the somedict.dict file.
> Done!

Có nghĩa là 1 file đuôi dz của stardict này là file nén (stardict có thể chạy được với file nén bằng định dạng dictZip (hình như thế) , bạn có thể làm theo cách nó hướng dẫn , nhưng mình thường dùng 7zip giải nén trực tiếp ra file dict luôn

Tiếp theo là định dạng file dsl của Lingvo

http://informationworker.ru/lingvo.en/index.html?page=lv_word_lang_dlg.htm

Nên dùng goldendict hoặc 1 dict nào đó hỗ trợ dsl để xem nó trình bày bố cục các tag thế nào

The source text of a DSL dictionary is a plain text file (*.txt), either in UNICODE or in ANSI encoding.

A DSL dictionary is made up of dictionary entries. An entry (card) has a heading (this can be a word or a phrase) and a body, which contains translations and comments. If a dictionary is added to the Lingvo Bookshelf, all of its card headings are displayed in the word list. Clicking a word or phrase in the word list, opens the dictionary card which displays the corresponding entry.

ABBYY Lingvo Multilingual Electronic Dictionary supports dictionaries not only in Russian but also in other European languages (for the full list of supported languages, see the Supported Languages section). The source and target languages of the dictionary must be specified in the DSL file. Therefore, at the beginning of each DSL file you need to specify the name of your dictionary (#NAME "dictionary name"), its source language (#INDEX_LANGUAGE "language name") and its target language (#CONTENTS_LANGUAGE "language name").

Ex.: If you wish to create an English-Russian dictionary named "General", type the following at the beginning of the DSL file:

```
#NAME "General"
#INDEX_LANGUAGE "English"
#CONTENTS_LANGUAGE "Russian"
```

If your dictionary consists of several files, use the #INCLUDE command to merge the files into one dictionary. Specify the full path to the file after a space or tabulation character. In the DSL language the back slash character "\" indicates that the character following it is text rather than a tag, so you have to use double slashes "\\" in the path to the file.
Ex.:

```
#INCLUDE "C:\\Dictionaries\\UniverseE.dsl"
```

Additionally, you can specify paths relative to the folder where the main DSL file (i.e. the file containing the #INCLUDE command) is located. In this case specify either the file name without the path or put a full stop (".") at the beginning of the path.
Ex.:

```
#INCLUDE "UniverseE.dsl"
#INCLUDE ".\\includes\\UniverseE.dsl"
```

All DSL tags have the following format: [a] (opening tag) or [/a] (closing tag).

[b], [/b] - boldfaced font

[i], [/i] - italics

[u], [/u] - underlined font

[c], [/c] - coloured (highlighted) font

[*], [/*] - the text between these tags is only displayed in full translation mode (see)

[mN] - sets the left paragraph margin. N is the number of spaces which will be used for the left-hand margin. N can be within the range from 0 to 9. The corresponding closing tag of the paragraph is [/m]. and left card margin.

[trn], [/trn] - translations zone.

[ex], [/ex] - examples zone.

[com], [/com] - comments zone.

[s],[/s] - multimedia zone (used to add pictures or sound files into a dictionary entries).

[url],[/url] - link to a Web page.

[!trs], [/!trs] - the text between these tags will not be indexed

[p], [/p] - labels (clicking a label displays its full text)

[lang][/lang] - the language of the word or phrase; use these tags to mark words in the card that are written in a language other than the target language. Words marked by [lang] tags will be indexed, and you will be able to find them when carrying out full-text searches, or translate them with a right-click. Parameters: language name or language ID, e.g. [lang id=1]. The name of the language must be enclosed in brackets, e.g. [lang name="Russian"]. See the list of supported DSL languages in the "Supported Languages" section.

[ref][/ref] - hyperlink to a card in the same dictionary (You can also use "<<" and ">>" to enclose the headword of the card to make a link).

[sub][/sub] - subscript

[sup][/sup] - superscript

Data dsl bạn có thể lấy từ trang này

http://traduko.lib.ru/dics_en_en.html

Từ ASCII đến Unicode

Đây là bài viết mình search trên mạng , vì từ điển của chúng ta là đa ngôn ngữ, nên sẽ dùng mã Unicode chứ không phải vni hay tcvn3 , các bạn nên có chút kiến thức về nó , bài viết này là bài viết thuộc loại dễ hiểu nhất mình từng biết

Từ ASCII đến Unicode.

kpham2@erols.com
(Xin cám ơn bạn Minh Sơn ở TP HCM đã dịch bài viết này từ Anh ra Việt).

Bài viết này là để giúp các bạn có trình độ máy tính trung bình hiểu được Unicode và UTF-8 rõ ràng hơn. Sau khi đọc xong, các bạn sẽ biết được lịch sử của Unicode, nó có các dạng thức nào, UTF-8 là gì và tại sao luôn đi đôi với Unicode.

Khi tổng hợp nên tài liệu này, để cho đơn giản, tôi đã bỏ qua nhiều khía cạnh hơi phức tạp của Unicode như các đề tài về mã tổ hợp, mã dựng sẵn. Nếu có thiếu sót, mong các bạn thông cảm. Thêm vào đó, bài viết không bàn về cách cài đặt/sử dụng Unicode font trong các hệ điều hành hay phần mềm. Về chuyện này, các bạn có thể tham khảo trang web của Lê Hoàn hay các thư trao đổi về Unicode.

Một vài điều cần lưu ý:

Trong bài viết, tôi chỉ dùng hệ thập lục phân (hệ 16) để chỉ giá trị của các mã. Ví dụ, khi tôi nói kí tự "a" có mã là 61, bạn phải hiểu rằng đây là 61 trong hệ thập lục phân (bằng 97 hệ thập phân). Lí do là trong các bảng mã, các mã thường có dạng thập lục phân chứ ít khi có dạng thập phân.

Ở cuối bài, tôi có một bảng mã Unicode cho các kí tự Việt nam các bạn tham khảo. Nếu muốn xem toàn bộ bảng mã Unicode

(dưới dạng file PDF) vào http://www.unicode.org. Trong đó, click "Code Charts" và bạn sẽ thấy nhiều "trang mã". Toàn bộ các kí tự tiếng Việt có thể được tìm thấy ở các trang Latin-1 Supplement, Latin Extend A và Latin Extend B, và Latin Extended Additional. Bạn có thể in các trang mã nếu muốn.

Cuối cùng, bạn có thể bỏ qua các phần mình đã biết và đi thẳng đến nơi tôi nói về UTF-16, UTF-8. Tuy nhiên, tôi cho rằng nếu bạn hiểu rõ hơn về các bảng mã ASCII và ANSI thì sẽ hiểu rõ hơn sự ra đời và phát triển của Unicode.

Một số định nghĩa hữu ích:

-Bảng mã: Một tập hợp nhiều kí tự khác nhau. Một ví dụ là bảng mã chuẩn ASCII (American Standard Code for Information Interchange - Mã chuẩn Hoa kỳ trong Trao đổi Thông tin) bao gồm 128 kí tự, phần lớn là các kí số, kí tự tiếng Anh, những ký tự đặc biệt và thông dụng như các dấu cộng, trừ, phần trăm... Unicode là một bảng mã chuẩn khác, gồm có hàng ngàn các kí tự gồm tiếng Anh và quốc tế bao gồm cả các kí tự Việt nam. Cũng có một vài bảng mã tiếng Việt (không chuẩn) như TCVN-ABC, VNI, VISCII, chúng chỉ có tối đa là 256 kí tự .

- Mã: Một số nguyên dương đại diện cho một kí tự trong một bảng mã. Mã của một kí tự thay đổi tùy theo bảng mã. Ví dụ, trong bảng mã tiếng Việt TCVN-ABC, kí tự "ầ" có mã C7. Trong bảng tiếng Việt VISCII, "ầ" có mã là A5. Trong bảng Unicode, "ầ" có mã là 1EA7 (=7847 thập phân). Lưu ý là mã của một kí tự cho thấy vị trí của kí tự trong bảng mã. Ví dụ, trong bảng Unicode, "ầ" nằm ở vị trí 7847 . Mỗi kí tự Unicode chỉ được "gắn" một mã duy nhất. Ví dụ, trong Unicode, bạn không thể tìm thấy kí tự "ầ" tại bất kỳ chỗ nào khác ngoài vị trí 7847. Các máy tính chỉ biết một kí tự qua mã của nó. Ví dụ, khi bạn đánh Unicode dùng một bộ gõ tiếng Việt và bạn muốn nhập chữ "ầ", bộ gõ tìm cách gửi mã 1EA7 (sau khi đã được mã hóa dưới dạng nhị phân) đến bộ xử lý trung ương của máy tính.

- Font Unicode: Một font được gọi là font Unicode khi nó cung cấp cấu hình của các kí tự trong bảng mã Unicode. Một font file (tập tin font) dùng mã của một kí tự để chỉ định cấu hình cho kí tự đó. Ví dụ, khi phải thể hiện kí tự "ầ" trên màn hình dùng font Arial, phần mềm sẽ lục tìm mã 1EA7 trong font file Arial.ttf và xác định cấu hình tương ứng. Nếu một font như VNI-Times không hổ trợ Unicode, nó sẽ không có cấu hình cho mã 1EA7 vì nó chỉ có mã lớn nhất là FF (=255 thập phân). Vì vậy, nó không thể hiển thị kí tự "ầ" và nó không được gọi là font Unicode. Tương tự như vậy, các font Arial, Times New Roman, Tahoma của các hệ điều hành như Windows 95 hoặc Windows 98 không có cấu hình cho các kí tự Unicode; do đó bạn phải "cập nhật" chúng bằng cách tải và cài đặt các font Unicode với các tên tương tự vào máy nếu bạn muốn đọc mail hay duyệt các web site dùng Unicode font.

- Chuỗi bit: Một chuỗi các số nhị phân, như 01100001. Do máy vi tính chỉ "đọc" được số nhị phân, dữ liệu phải được chuyển đổi thành các chuỗi bit trước khi được nhập vào máy. Mỗi kí số trong một số thập lục phân luôn được biểu diễn bằng bốn 4 số nhị phân. Ví dụ, 6 = 0110, 1 = 0001, F = 1111, 7 = 0111, 61 = 01100001, 7F=01111111.

- Mã hóa (encoding): Cách biểu diễn một kí tự trong dạng một chuỗi bit. Tùy theo cách mã hóa, một kí tự có thể được biểu diễn khác nhau.

"UTF-16" là một kiểu mã hóa các kí tự Unicode trong đó mỗi kí tự được biểu diễn dưới dạng một chuỗi 16-bit tương đương với giá trị của mã. Ví dụ, trong UTF-16, "ầ" được mã hoá thành một chuỗi 16-bit: 0001111010100111 (= 1EA7), tương đương với mã gốc của "ầ" trong bảng Unicode.

"UTF-8" là một kiểu mã hóa khác cho các kí tự Unicode, trong đó mỗi kí tự được biểu diễn dưới dạng MỘT hay NHIỀU chuỗi 8-bit, có thể KHÔNG tương đương với mã gốc. Ví dụ, trong UTF-8, "ầ" được mã hóa thành ba chuỗi 8-bit (cũng có thể gọi là một chuỗi 24 bit) 111000011011101010100111 (= E1BAA7) không tương đương với mã gốc là 1EA7. Tại sao cần UTF-8. Chúng ta sẽ biết sau.

- Giải mã: Sau khi hệ điều hành nhận được một kí tự (ví dụ đọc từ một file) đã được mã hóa, nó phải giải mã để lấy lại mã gốc của kí tự trong bảng mã trước khi vào font file để tìm cấu hình và thể hiện kí tự trên màn hình. Một font file chỉ dùng các mã gốc chứ không dùng dạng đã mã hóa.

Hệ ASCII/ANSI: các hệ điều hành chỉ dùng các bảng mã ASCII hay ANSI. Ví dụ: Windows 95 dùng bảng mã ANSI. Các hệ ASCII và ANSI luôn luôn dùng một đơn vị dữ liệu là 8 bit (1 byte).

QUÁ TRÌNH PHÁT TRIỂN: từ ASCII đến ANSI cho đến Unicode.

1. Bảng mã ASCII: 7-bit, cho phép 128 mã (2 mũ 7) Còn có tên khác là ISO 646-IRV. ASCII là bộ mã đầu tiên lúc máy tính được
phát minh
Mã cho phép: từ 0 đến 7F
Mã nhỏ nhất: 0, dùng cho kí tự NUL (null: trống trơn, không có gì).
Mã lớn nhất 7F (=thập phân 127, =nhị phân 01111111). Được dùng cho phím DEL (delete-xoá).
(lưu ý: mặc dù đơn vị dữ liệu là 8 bit, chỉ có 7 bit cuối được dùng,)

Ví dụ: Trong bảng ASCII, kí tự "a" có mã là 61.
Khuyết điểm: chỉ có 128 kí tự được cho phép. Mọi người cần nhiều mã hơn, nhất là sau khi hệ DOS và máy tính cá nhân xuất hiện. Vì vậy, người ta phải nghĩ ra bộ mã ANSI.

2. Bảng mã ANSI : 8-bit, là bảng mã ASCII mở rộng; cho phép 256 mã (2 mũ 8).
Các tên khác: ISO-8859-1, LATIN-1.
Mã cho phép: từ 0 đến FF
Mã nhỏ nhất: 0, dùng cho kí tự NUL.
Mã lớn nhất 255 = FF (=thập phân 255, =nhị phân 11111111) .
(lưu ý: tất cả 8 bit trong đơn vị dữ liệu được dùng)

Ví dụ: trong bảng ANSI, kí tự "ô" của tiếng Việt có mã là F4. (các bộ mã tiếng Việt đều dựa trên ANSI với nhiều sửa đổi)
Lưu ý: 128 kí tự đầu tiên (các mã từ 0...7F) giống nhau trong ASCII và ANSI.

Ví dụ, ký tự "a" có mã là 61 trong cả hai bảng ASCII và ANSI. Nói cách khác, ASCII là tập con của ANSI.

Ưu điểm: số lượng mã cho phép đã được tăng đến 256. Do đó, bây giờ bảng mã có chỗ cho các kí tự khác bên cạnh tiếng Anh.
Khuyết điểm: Vẫn chưa đủ chỗ cho các kí tự quốc tế. (Tàu, Hàn Quốc, Ä Rập, Do Thái...,quá nhiều!) Vì vậy, người ta phát minh ra Unicode 16-bit.

3. Bảng mã Unicode 16-bit: Cho phép 65536 mã. (2 mũ 16)
Các tên khác: ISO-14646, UCS-2.
Mã cho phép: từ 0 đến FFFF Mã nhỏ nhất: 0, dùng cho NUL
Mã lớn nhất 65535 = FFFF (= thập phân 65535, = nhị phân 1111111111111111)

Ví dụ: trong bảng Unicode, kí tự "â`" của tiếng Việt có mã là 1EA7.
Lưu ý: 256 kí tự đầu tiên (các mã từ 0...255= FF) giống nhau trong ANSI và Unicode. Ví dụ, ký tự "a" có mã là 61 trong cả ba bảng ASCII và ANSI và Unicode. Nói cách khác, ANSI (cũng như ASCII) là tập con của Unicode.

Ưu điểm: đủ chỗ chứa toàn bộ các kí tự của các dân tộc trên thế giới.
Khuyết điểm: Hầu hết các máy tính vẫn còn dùng bộ mã ASCII, do đó chúng không nhận ra các mã lớn hơn 7F. Và còn một vấn đề lớn hơn là, các hệ ASCII và ANSI, vốn chỉ xử lý dữ liệu theo từng chuỗi 8-bit, sẽ lầm lẫn khi xử lý các kí tự Unicode được mã hóa dưới dạng 16-bit (UTF-16). Các hệ ASCII/ANSI sẽ diễn dịch MỘT kí tự Unicode 16-bit thành HAI kí tự 8-bit. Ví dụ, kí tự "a" dạng 16-bit sẽ được dịch thành HAI kí tự: kí tự thứ nhất là NUL (00000000), và kí tự thứ hai là kí tự ASCII "a" (01100001). Chẳng hạn, khi bạn muốn thể hiện hàng chữ : "ABCDEF" được mã hóa UTF-16, có khả năng bạn sẽ nhìn thấy " A B C D E F" trên màn hình. (trên màn hình, các kí tự NUL có thể được thể hiện thành các ô trống hay là các ô vuông, tùy theo máy).
Vấn đề này cần phải được giải quyết. Chúng ta vẫn muốn dùng bảng mã Unicode nhưng cần mã hoá các kí tự theo cách nào đó mà các hệ ASCII có thể nhận ra các kí tự của chúng ta. Cách mã hoá UTF-16 rõ ràng là có vấn đề cho các hệ điều hành phổ biến hiện nay vẫn đang dùng chuẩn ASCII/ANSI. Đó là lý do người ta sáng chế ra cách mã hoá UTF-8.

4. Nguyên tắc mã hoá UTF-8:
- Một kí tự Unicode sẽ được mã hóa thành một hay nhiều chuỗi 8-bit để các hệ ASCII hay ANSI có thể nhận diện.
- Để tương thích với ASCII, các kí tự Unicode thuộc bảng mã ASCII (mã từ 0 đến 7F) được mã hóa thành một chuỗi 8-bit tương đương với giá trị nhị phân của mã. Vì bảng ASCII chỉ có thuần các kí tự tiếng Anh, điều này cũng có nghĩa là các hệ ASCII có thể đọc các văn bản tiếng Anh viết bằng Unicode UTF-8 một cách dễ dàng, không cần phải chuyển đổi gì.
- Tất cả các kí tự Unicode có mã lớn hơn 7F được mã hoá thành HAI hoặc BA chuỗi 8-bit (byte) phù hợp với nguyên tắc trong bảng phía dưới.
- Trong UTF-8, byte đầu tiên của một kí tự Unicode sẽ chỉ định có bao nhiêu byte đi kèm theo dành cho kí tự đó. Như vậy nếu một hệ ASCII/ ANSI sau khi đọc được byte thứ nhất của một kí tự UTF-8 thì sẽ biết có bao nhiêu byte đi kèm cho kí tự đó. Điều này giúp cho nó trong việc giải mã (để lấy trở lại mã Unicode) cho kí tự.

Dưới đây là hai bảng mã hoá UTF-16 và UTF-8 cho các kí tự Unicode. Trong cáccbảng, một chữ "x", "y" hoặc "z" có thể là một bit 0 hoặc một bit 1.

Bảng A: Cho mã có giá trị từ 0 đến 7F (các kí tự ASCII):

mã	UTF-16		UTF-8
	byte 1	byte 2	
0-7F	00000000	0xxxxxxx	0xxxxxxx

Bảng B: cho mã từ hex 80 trở lên:

mã	UTF-16		UTF-8		
	byte 1	byte 2	byte 1	byte 2	byte 3
80-7FF	00000yyy	yyxxxxxx	110yyyyy	10xxxxxx	
800-FFFF	zzzzyyyy	yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx

Theo bảng A::
- Nếu mã NHỎ HƠN hoặc BẰNG 7F thì được mã hoá thành 8-bit tương đương với dạng nhị phân của mã.

Theo bảng B:
- Nếu mã LỚN HƠN 7F và NHỎ HƠN hoặc BẰNG 7FF thì được mã hoá thành 2 chuỗi 8-bit.
- Nếu mã LỚN HƠN 7FF thì được mã hoá thành 3 chuỗi 8-bit.

Ví dụ: Mã hoá kí tự Unicode tiếng Việt "ầ" (mã = 1EA7) dùng UTF-8:

1) Đầu tiên viết mã thành 1 chuỗi 16-bit (UTF-16): 0001111010100111 tương đương với 1EA7.
2) Cắt chuỗi 16-bit thành hai byte: byte 1 là: 00011110 và byte 2 là: 10100111.
3) 1EA7 lớn hơn 7FF và nhỏ hơn FFFF. Theo bảng trên, dùng dòng cuối cùng để chuyển đổi (nghĩa là dạng mã hóa UTF-8 của bạn cho kí tự "ầ" sẽ có 3 chuỗi 8-bit (3-byte) .
4) Đối chiếu với byte 1 và byte 2 trong dòng cuối của cột UTF-16, bạn sẽ có: zzzz = 0001; yyyyyy = 111010; và xxxxxx = 100111.
5) Đối chiếu với byte 1 và byte 2 trong dòng cuối của cột UTF-8, bạn sẽ có dạng UTF-8:
byte 1 là: 1110zzzz = 11100001. (=E1)
byte 2 là: 10yyyyyy = 10111010 (= BA)
byte 3 là: 10xxxxxx = 10100111 (=A7).
Tổng hợp lại, kí tự "ầ" đã được mã hóa dưới dạng UTF-8 là: E1BAA7.

Lưu ý rằng bây giờ bạn có 3 byte cho kí tự ầ, khác với kí tự gốc chỉ có 2 byte. Nếu bạn theo nguyên tắc trên, bạn có thể viết

các trình mã hoá/giải mã UTF-8 cho hệ thống của mình.

Thêm vài ví dụ UTF-8

kí tự/ mã	UTF-8
a 97	97
A. 7840	225, 186,160;
A(` 7856	225, 186,176;
E^` 7872	225, 187,128;
O^’ 7888	225, 187, 144;
O*~ 7904	225, 187, 160

Bảng Unicode cho các kí tự Việt Nam.

225 a'
224 a`
7843 a?
227 a~
7841 a.
7855 a(' a(` a(? a(~.
7857
7859
7861
7863
7845 a^' a^` a^? a^~ a^.
7847
7849
7851
7853
250 u' u` u? u~ u.
249
7911
361
7909
7913 u*' u*` u*? u*~ u*.
7915
7917
7919
7921
233 e' e` e? e~ e.
232
7867
7869
7865
7871 e^' e^` e^? e^~ e^.
7873
7875
7877
7879
243 o' o` o? o~ o.
242
7887
245
7885
7889 o^' o^` o^? o^~ o^.
7891
7893
7895
7897
7899 o*' o*` o*? o*~ o*.
7901
7903
7905
7907
237 i' i` i? i~ i.
236
7881
297
7883
253 y' y` ...
7923

7927
7929
7925
259 a(a^ u* e^ o* o^
226
432
234
417
244
273 d-
193 A' A` A? A~ A.
192
7842
195
7840
7854 A(' A(` A(? A(~ A(.
7856
7858
7860
7862
7844 A^' A^` A^? A^~ A^.
7846
7848
7850
7852
218 U' U` U? U~ U.
217
7910
360
7908
7912 U*' U*` U*? U*~ U*.
7914
7916
7918
7920
201 E' E` E? E~ E.
200
7866
7868
7864
7870 E^ E^ E^ E^ E^
7872
7874
7876
7878
211 O' O` O? O~ O.
210
7886
213
7884
7888 O^' O^` O^? O^~ O^.
7890
7892
7894
7896
7898 O*' O*` O*? O*~ O*.
7900
7902
7904
7906
205 I' I` I? I~ I.
204
7880
296
7882
221 Y' Y` Y? Y~ Y.
7922
7926
7928
7924

258 A(A^ U* E^ O* O^
195
431
202
416
212
208 D-

Ngôn ngữ lập trình và các tiện ích cần chuẩn bị

Vì từ điển của chúng ta là từ điển đa ngôn ngữ thế nên việc cần thiết là phải hỗ trợ unicode , ngoài ra nên hỗ trợ sắp xếp đa ngôn ngữ (không nhất thiết phải có).
Mình chọn java (các bản spdict trước 6.0 thì dùng c#) để lập trình , c# , vb.net cũng được , 2 cái này thoả mãn cả hai tính năng trên, các ngôn ngữ dùng framework hình như đều hỗ trợ . Còn không các bạn có thể dùng vb (lấy control unicode của bên caulacbovb.net) hoặc delphi , vc++ cũng được, tuỳ , cái phần sắp xếp đa ngôn ngữ không nhất thiết phải có vì nó chỉ làm thay đổi phần hiển thị danh sách một chút , không đáng kể , cái này chỉ mang tính chất thói quen người dùng thôi.

Ngoài ra , để làm việc với file dict.tab các bạn còn cần một công cụ có khả năng hiển thị file text hàng chục mb thậm chí hàng trăm mb với tốc độ nhanh , việc này notepad thậm chí word cũng không làm được. Phải dùng 1 số công cụ như notepad2 , notepad++ , EmEditor , EditPlus ...
notepad++ nhiều tính năng hơn, nhưng hiển thị unicode một số ký tự thành ô vuông, mấy cái kia không free và phải cài vì vậy mình quyết định chọn notepad2

Như vậy là xong, chúng ta bắt đầu nghiên cứu tiếp các bài sau.

Bắt đầu làm việc với file dữ liệu từ điển

Để có thể code từ điển thì các bạn phải thông thạo đọc và ghi 2 loại file nhị phân và văn bản .

Trong file văn bản thì cần biết cách đọc dữ liệu unicode (spdict lưu dạng UTF-8) và đọc tuần tự từng dòng , từng ký tự ….

Đối với file nhị phân thì cần chú ý:
+position (cho biết vị trí con trỏ văn bản hiện hành)
+seek : nhảy đến các vị trí trong văn bản
+setlength : định lại kích thước của file.
+Đọc mảng byte với độ dài cho sẵn, đọc 1 số kiểu nhị phân short (2 byte) , integer(4 byte)

Progressbar và thread

Để chạy các quá trình convert và tìm kiếm nâng cao thì thanh hiển thị tiến trình progressbar là rất cần thiết ,để chạy nó thì các bạn search google hoặc lên trang sun có ví dụ về cách thức sử dụng control này (swing control) , ngoài ra thì để progressbar và chương trình không bị đơ khi chạy quá trình tìm , convert , ta cần dùng đến thread.

Lấy đường dẫn của thư mục hiện hành (riêng cho java)

Ở các chương trình c# , vb , vb.net chỉ cần application startup path là ra đường dẫn nhưng java thì không thế . Java dùng hàm này để lấy đường dẫn thư mục hiện tại :

System.getProperty("user.dir");

Nhưng nó chỉ đúng với môi trường window , với môi trường linux thì nó luôn ra thư mục home như máy mình là :

/home/tienlbhoc

Để có thể lấy được đường dẫn trên cả win và linux thì các bạn dùng đoạn mã sau :

```
        URL link = this.getClass().getProtectionDomain().getCodeSource().getLocation();
//lấy đường dẫn class hiện tại dạng url
        File i = new File(link.toURI());
        duongDanChinh = i.getParent();//convert ra dạng path bình thường
```

Chỉ cho phép ứng dụng chạy 1 lần duy nhất (single instance)

Cái này khá quan trọng vì nếu chạy 2 từ điển cùng 1 lúc đọc cùng 1 file từ điển thì từ điển này sửa dữ liệu, từ điển kia chưa thay đổi theo sẽ gây lỗi chương trình. Có nhiều cách để làm việc này nhưng đây là 1 trong những cách ngắn nhất.

```
import java.net.ServerSocket;
import javax.swing.JOptionPane;
import javax.swing.JFrame;
import java.io.IOException;
import java.net.BindException;

class SingleInstance {

   public static ServerSocket serverSocket;
   public static String errortype = "Access Error";
   public static String error = "Application already running.....";
   public static void main(String as[]) {
      try {//creating object of server socket and bind to some port number
         serverSocket = new ServerSocket(15486);
////do not put common port number like 80 etc.
////Because they are already used by system
         JFrame jf = new JFrame();
         jf.setVisible(true);
         jf.setSize(200, 200);
         jf.setDefaultCloseOperation(jf.EXIT_ON_CLOSE);
      } catch (BindException exc) {
         JOptionPane.showMessageDialog(null, error, errortype, JOptionPane.ERROR_MESSAGE);
         System.exit(0);
      } catch (IOException exc) {
         JOptionPane.showMessageDialog(null, error, errortype, JOptionPane.ERROR_MESSAGE);
         System.exit(0);
      }
   }
}
```

Các chức năng của từ điển

So sánh và sắp xếp theo ngôn ngữ

Java hỗ trợ sắp xếp nhiều ngôn ngữ trên thế giới (trong đó có tiếng việt) , để sắp xếp , so sánh ta cần phải tạo 1 class kế thừa class Comparator (class so sánh trong java)

```
import java.text.Collator;
import java.util.Comparator;
import java.util.Locale;

/**
 *
 * @author tien
 */
public class LangComparator implements Comparator {

    Collator collator;
    Locale locale;

    public LangComparator(String lang) {
        locale = new Locale(lang);
        collator = Collator.getInstance(locale);
    }

    public int compare(Object emp1, Object emp2) {
        return collator.compare((String) emp1, (String) emp2);
    }

    public int SoSanh(String emp1, String emp2) {
        if (emp1 == null) {
            emp1 = "";
        }
        if (emp2 == null) {
            emp2 = "";
        }
        return collator.compare(emp1, emp2);
    }

    public int compareThuong(String emp1, String emp2) {
        return collator.compare(emp1.toLowerCase(locale), emp2.toLowerCase(locale));
```

```
    }
}
```

Để lấy danh sách các bảng mã sắp xếp thì các bạn tham khảo đoạn code sau:

```
String[] mangSort = Locale.getISOLanguages();
Locale l, l1 = new Locale("vi");
cbbMaSapXep.removeAllItems();
for (int i = 0; i < mangSort.length; i++) {
   l = new Locale(mangSort[i]);
   cbbMaSapXep.addItem(l.getLanguage() + " : " + l.getDisplayLanguage(l1));
}
cbbMaSapXep.setSelectedItem("en : Tiếng Anh");
//nếu l1 là en thì tên của nó sẽ là “en : English” chứ không phải “en : Tiếng Anh”
```

Các quá trình sắp xếp với mảng , danh sách liên kết với langComparator ở trên thì trong java đã có class sẵn , mà nếu bí quá không biết dùng thì tự tạo code sắp xếp cũng được (lúc sắp xếp thì so sánh 2 string bằng class langComparator là được)

Phát âm cho từ điển

Dùng thư viện phát âm free java text to speech của java http://freetts.sourceforge.net/ để tải thư viện mới nhất + mã nguồn hướng dẫn sử dụng

Hiển thị dữ liệu có định dạng màu sắc + chuyển tiếp từ

(trong phần nghĩa của từ được tra)

Java hỗ trợ control jtextpane , hỗ trợ mã html (tuy nhiên chỉ hạn chế thôi, html 3.2 thì phải) , nhưng cũng thừa đủ dùng rồi , để sử dụng nó thì cần set 2 thuộc tính sau :

```
jTextPane1.setContentType("text/html");
jTextPane1.setEditable(false);
```

Sau đó , muốn convert văn bản thì chỉ việc settext cho nó là được , ví dụ hiển thị chữ Tiến in đậm

```
jTextPane1.setText(“<b>Tiến</b>”);
jTextPane1.setCaretPosition(0); //cuốn về đầu trang sau khi hiển thị
```

nếu không thạo html có thể dùng 1 trình soạn web nào đó như microsoft web expression , dreamwave ….

Và ta cũng có thể dùng hyper link để tạo chuyển tiếp từ cho từ điển :

Đầu tiên add sự kiện (even lắng nghe quá trình kích hyperlink

```
jTextPane1.addHyperlinkListener(new HyperlinkListener() {
public void hyperlinkUpdate(HyperlinkEvent e) {
```

```
        if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {
            TraAllVaHienThi(e.getDescription());// e.getDescription() là đoạn text được lấy về , chính là từ cần tra
        }
    }
});
```

Bây giờ chỉ cần hiển thị html có nội dung dạng như sau , với đoạn code trên sẽ bắt được từ xin chào

```
<a href="xin chào" style="text-decoration: none">xin chào</a>
```

Tra từ qua clipboard

Đây là một tiện ích tra từ trong ứng dụng khác , chỉ việc bôi đen text và gõ Ctrl+C , code cực kỳ ngắn thế này thôi , đại thể là cho một cái timer thời gian = 200 (nhỏ hơn cũng được) , cứ sau khoảng thời gian đó kiểm tra xem clipboard có thay đổi gì không, nếu thay đổi thì lấy text còn không thì thôi , đây là code tạo timer và lấy clipboard :

```
Timer t = new Timer(200, new ActionListener() {
public void actionPerformed(ActionEvent e) {
    try {
        if (jcbClipboard.isSelected() == true) {
            tk = Toolkit.getDefaultToolkit().getSystemClipboard().getContents(null);
            if (tk != null && tk.isDataFlavorSupported(DataFlavor.stringFlavor)) {
                String tuMoi = (String) tk.getTransferData(DataFlavor.stringFlavor);
                if (tuMoi.length() < 200 && tuMoi.equals(tuCu) == false) {
                    jTextField1.setText(tuMoi);
                    tuCu = tuMoi;
                    jTextPane1.setText(frmMain.TraAll(jTextField1.getText()));
                    jTextPane1.setCaretPosition(0);
                }
            }
        }
    } catch (Exception exception) {
    }
}
});
t.start();
```

Thuật toán làm từ điển

Đo tốc độ thuật toán

Có nhiều cách khác nhau , có phần mềm riêng để làm , nhưng mình nghĩ cái này dễ làm nhất:

```
long c = Calendar.getInstance().getTimeInMillis();
//đoạn chương trình cần kiểm tra tốc độ
c = Calendar.getInstance().getTimeInMillis() - c;
//c sẽ có giá trị mili giây của thời gian đoạn chương trình đó
```

Một số thuật toán làm từ điển

He he, quá trình tìm thuật toán làm từ điển của mình nan giải lắm . Đầu tiên là mấy bài làm từ điển bằng dos, dùng cây nhị phân , một thuật toán từ điển khá nhanh và hầu hết sinh viên mấy năm đầu cntt làm từ điển đều đi theo hướng này (chắc có code sẵn) , tuy nhiên cây sẽ bị mất đối xứng trong quá trình thêm xoá, code hơi phức tạp và dễ lỗi. Theo mình ai mà làm từ điển theo kiểu cây thì nên đi theo các loại cây cân bằng như cây AVL , cây cờ bạc (cây đỏ đen) , các bạn search trên http://vi.wikipedia.com sẽ có những giới thiệu rất cơ bản dễ hiểu về các loại cây này. Mình không thạo về cây lắm, nhưng cây có một ưu điểm hơn định dạng của mình là tốc độ thêm xoá nhanh lắm , nhưng cũng có nhược điểm là để nhảy đến một vị trí bất kỳ (vấn đề sống còn trong kỹ thuật load danh sách) thì cần phải cải tiến nhiều đó , hiện mình chưa biết cách nào để nhảy đến vị trí n nhanh nhất cả .

Tiếp theo là dùng bảng băm để làm từ điển, cái này tìm trên net rất nhiều người bảo lạc việt dùng cái này (tin vịt vỉa hè) , đặc điểm của cái này là từ từ khoá 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, nhanh hay chậm tuỳ thuộc vào hàm băm , search wikipedia để biết thêm chi tiết.

Còn một cách nữa đó là xài cơ sở dữ liệu access , xml , sqlite (nghe nói thằng sqlite này nhỏ , đa nền tảng và mạnh hơn access) ... có từ điển echip làm bằng access đó, nếu dùng nó thì rất dễ code , nhiều tính năng , nhưng nếu có định dạng từ điển riêng thì vẫn thích hơn vì mình có thể quản lý được linh hoạt hơn

Thuật toán tìm kiếm từ điển của mình : tìm kiếm nhị phân , đừng nhầm lẫn với cây nhị phân nhé . Khi các bạn đọc ebook này không nhất thiết phải đi theo hướng của mình giống như trước đây , mọi người toàn bảo mình làm bằng bảng băm đó .

Chuẩn dict.org

Đây là bài viết của anh Trần Bình An, admin tudientiengviet.net chính nơi chúng ta lấy dữ liệu về xài, cũng chính bài viết này mà chuẩn dict.org đã được biết đến nhiều ở việt nam và các từ điển multidictionary, powerclick , jtranslator mới xuất hiện . Từ điển của mình , chuẩn SPDict cũng là nâng cấp 3 lần liên tục của định dạng ban đầu này , nhưng bây giờ định dạng SPDict thấy nó giống mảng con trỏ hơn.

Tự xây dựng một ứng dụng từ điển đơn giản

Việc học ngoại ngữ hiện này đã trở thành nhu cầu không thể thiếu đối với rất nhiều người. Và vật dụng cần nhất khi học ngoại ngữ đó chính là quyển từ điển. Cũng như các bạn, khi học ngoại ngữ tôi cũng phải dùng từ điển. Tuy nhiên, chắc hẳn các bạn cũng như tôi sẽ cảm thấy rất vất vả khi phải tra từ trên từ điển . Và giải pháp đáng giá là sử dụng các ứng dụng từ điển trên máy vi tính. Mặc dù hiện nay các ứng dụng từ điển này đã có nhiều nhưng vốn là dân tin học, tôi đã quyết định tự xây dựng cho mình một ứng dụng từ điển riêng. Tôi sẽ hướng dẫn các bạn làm một ứng dụng từ điển cho mình, và chắc chắn các bạn sẽ tận hưởng được cảm giác vui sướng như tôi mỗi khi sử dụng từ điển do chính mình làm ra.

I. Cơ sở dữ liệu:

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 nghìn từ. Thật may mắn, www.dict.org đã xây dựng một format từ điển rất dễ sử dụng, Dạng format này đã được một số cá nhân sử dụng để xây dựng những bộ từ điển khá lớn. Dict format được mô tả như sau: toàn bộ cơ sở dữ liệu được chứa trong 2 file, một file chứa nghĩa của từ và một file index. File index bao gồm tên từ, vị trí nghĩa của từ bắt đầu trong file chứa nghĩa và độ dài của nghĩa. Vị trí bắt

đầu và độ dài của nghĩa được mã hoá theo cách như sau: Sử dụng 64 chữ cái:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

chữ cái A tương đương số 0, chữ cái B tương đương số 1 v.v… Giữa từ, vị trí bắt đầu và độ dài nghĩa phân cách nhau bởi ký tự tab (ASCII 9). Mỗi dòng trong file index chứa dữ liệu của một từ. Các dòng phân cách nhau bởi ký tự xuống dùng (ASCII 10).

Ví dụ như trong file index của từ điển Đức Việt có một dòng như sau:

Abdeckung kbpP D3

Như vậy nghĩa của từ Abdeckung trong file chưa nghĩa sẽ bắt đầu tại offset kbpP (theo mã 64 ký tự) và có độ dài là D3.

Việc chuyển từ mã cơ số 64 về cơ số 10 được thực hiện như sau:

Đối với vị trí bắt đầu: kbpP. Ta có k ở mã cơ số 64 = 36 ở cơ số 10, b = 27, p = 41, P = 15. Như vậy chuyển sang cơ số 10, mã kbpP có giá trị là: 36*64^3 + 27*64^2 + 41*64^1 + 15*64^0 = 9550415

Đối với độ dài nghĩa: D3. Ta có D = 3, 3 = 55. Như vậy chuyển sang cơ số 10, mã D3 ở cơ số 64 có giá trị là: 247

File index được sắp xếp để giảm bớt thời gian tìm kiếm. Việc mã hoá theo cơ số 64 như trên giúp cho kích thước file index sẽ giảm xuống rất nhiều khi so với khi không mã hóa.

Còn cấu trúc của file chứa nghĩa gồm các phần như sau:

@headword
* tu loai (noun, verb...)
- dinh nghia 1
= cau vi du cho dinh nghia 1 + nghia cua cau do
- dinh nghia 2
= cau vi du cho dinh nghia 2 + nghia cua cau do
* tu loai
- dinh nghia 3
Nghĩa của mỗi từ gồm một phần như trên, các nghĩa của mỗi từ nối tiếp nhau liên tục.

Như vậy, các bạn đã hiểu cách thức và hoàn toàn có thể xây dựng được cho mình các bộ từ điển riêng rồi. Tuy nhiên, công việc nhập dữ liệu lại không hề đơn giản một chút nào. Nhưng, lại một lần nữa, chúng ta thật may mắn vì đã có một số bạn bỏ công ra nhập sẵn cho chúng ta một số bộ từ điển thông dụng rồi. Các bạn có thể tham khảo thêm tại: http://www.ttdomain.net/ttdownload/, http://www.informatik.uni-leipzig.de/~duc/Dict/, http://huybien.vze.com.. Ngoài ra còn rất nhiều bộ từ điển chuyên dụng khác nữa, các bạn có thể tham khảo thêm ở các địa chỉ trên hoặc tại www.dict.org.

II. Xây dựng chương trình:

Ở đây tôi xin trình bầy cách sử dụng ngôn ngữ Visual C++ 6.0 và thư viện MFC. Các bạn hoàn toàn có thể dễ dàng sử dụng các ngôn ngữ khác để làm. Trong khuôn khổ một bài báo, tôi chỉ xin đưa ra những phần chủ yếu nhất. Các phần như thiết kế giao diện, bố trí giao diện v.v… tôi xin dành cho các bạn tự sáng tạo.

1. Các thành phần giao diện cơ bản:

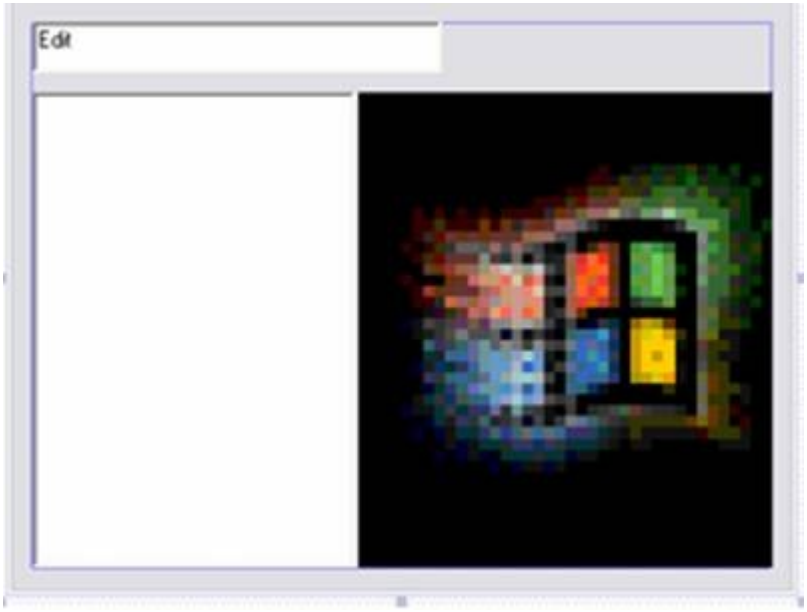
- Edit Box: dùng để nhập từ;Gán biến: Variable name: m_word, Category: Value, Type: CString

- WebBrowser dùng để hiện nghĩa của từ. Việc sử dụng WebBrowser chỉ nhằm mục đích hiển thị nghĩa trực quan và sinh động hơn bằng cách xử lý chuỗi (sẽ được đề cập sau này). Bạn hoàn toàn có thể thay thế bằng một điều khiển Edit. Bạn có thể thêm điều khiển ActiveX Web Browser vào ứng dụng của mình bằng cách chọn Project -> Add to Project -> Components And Controls, chọn trong thư mục Registered ActiveX Controls điều khiển Microsoft Web Browser. Cài biến cho điều khiển: Variable name : m_wordmean;

- Listbox, dùng để hiện danh sách từ. Cài biến cho điều khiển: Variable name: m_wordlist; Category: Control;

- Listbox, dùng để lưu trữ dữ liệu về từ. Cài biến cho điều khiển: Variable name: m_worddata, Category:Control;

Giao diện chương trình sẽ như sau:



2. Mã chương trình:

- Nạp dữ liệu vào các list box: bạn đặt phần mã này ở sự kiện WM_OnInitDialog() để dữ liệu được nạp ngay từ lúc khởi động chương trình. Ở đây, bạn thay tên file index bằng tên file tương ứng với từ điển bạn sử dụng:

```
FILE *inFile;
inFile = fopen ("mydic.index","r");
if (inFile == NULL)
{
	MessageBox ("Cannot open index file");
}
else
{
	char * line;
	char lineBuf[100];
	line = (char *) lineBuf;

	m_wordlist.ResetContent();
	m_worddata.ResetContent();
	CString word = "";
	CString sWord = "";
	CString sData = "";
	while (!feof(inFile))
	{
		fgets(line,99,inFile);
		if (strlen(line)>=2)
		{
			word = line;
			int pos = word.Find("\t",0);
			sWord = word.Left(pos);
			sData = word.Mid(pos+1,word.GetLength()-pos-1);
```

```
                        if (sData.Find("\n",0) > 0){
                                sData = sData.Left(sData.GetLength()-1);
                        }
                        if (sWord.GetLength()>=1)
                        {
                                m_wordlist.AddString(sWord);
                                m_worddata.AddString(sData);
                        }
                }
        }
        fclose(inFile);
```

- Hàm chuyển từ mã cơ số 64 sang cơ số 10:

```
int GetDemicalValue (CString str)
{
        CString base64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
        int decValue = 0;
        int len = str.GetLength();
        for (int i = 0; i<len; i++)
        {
                int pos = base64.Find(str.GetAt(i),0);
                decValue += (int)pow(64,len-i-1)*pos;
        }
        return decValue;
}
```

- Hàm xử lý chuỗi ký tự nghĩa. Như đã đề cập ở trên, ở đây tôi sử dụng điều khiển Web Browser để hiển thị nghĩa cho thêm phần sinh động. Hàm này có tác dụng xử lý chuỗi ký tự nghĩa bằng cách thêm các tag HTML để nghĩa được thể hiện sinh động hơn ví dụ như: các nghĩa con thì in đậm, chữ đỏ, các ví dụ thì in nghiêng, chữ xanh v.v….

```
CString ChangeStyle(CString wordmean)
{
        CString meaning = wordmean;
        meaning = meaning.Right(meaning.GetLength()-1);
        int pos = meaning.Find("\n",1);
        meaning.Insert(pos,"</b>");
        meaning = "<b>" + meaning;
        meaning.Replace("\n","<br>");
        meaning.Replace("{","<font color=\"#FF0000\"><b>");
        meaning.Replace("}","</b></font>");
        meaning.Replace("[","<font color=\"#FF0000\"><b>");
```

```
	meaning.Replace("]","</b></font>");
	meaning.Replace('+',' ');
	return meaning;
}
```

- Hàm lấy nghĩa của từ: hàm này có tác dụng đọc từ file chứa nghĩa để lấy nghĩa của từ, sau đó xử lý chuỗi nghĩa rồi ghi ra file temp.htm.Nếu lấy nghĩa thành công, hàm trả về giá trị TRUE, nếu lấy nghĩa không thành công, hàm trả về giá trị FALSE. Bạn thay tên file mydict.dict bằng tên của file từ điển tương ứng.

```
BOOL CXDictDlg::GetMeaning ()
{
	CFile f;
	CString meaning="";
	if (f.Open("mydic.dict",CFile::modeRead) == FALSE)
	{
		meaning = "Can not open database file!";
	}
	else
	{
		CString sOffLen;
		m_worddata.GetText(m_wordlist.GetCurSel(),sOffLen);
		int pos = sOffLen.Find("\t",0);
		CString sOff = sOffLen.Left(pos);
		CString sLen = sOffLen.Right(sOffLen.GetLength()-pos-1);
		int iOff = GetDemicalValue(sOff);
		int iLength = GetDemicalValue(sLen);

		int temp = f.Seek(iOff,CFile::begin);
		char buff[64];
		DWORD dwRead;
		do
		{
			if (iLength>=64)
				dwRead = f.Read(buff,64);
			else
				dwRead = f.Read(buff,iLength);
			iLength -= dwRead;
			CString stemp = buff;
			stemp = stemp.Left(dwRead);
			meaning += stemp;
		}while (iLength>0);
		f.Close();
```

```
	}
	meaning = ChangeStyle(meaning);
	CString strHtml("");
	strHtml += "<html>\n<head>\n";
	strHtml += "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">\n";
	strHtml += "</head>\n<body>\n";



	strHtml += meaning + "\n</body>\n</html>";
	CFile f2;
	if (f2.Open("temp.htm",CFile::modeCreate | CFile::modeWrite) == FALSE){
		MessageBox("Cannot write meaning file!","Error!");
		return 0;
	}

	f2.Write(strHtml,strHtml.GetLength());
	f2.Close();

	return 1;
}
```

- Hiển thị nghĩa của từ, bạn cài đoạn mã này vào sự kiện DoubleClick của điều khiển list box.

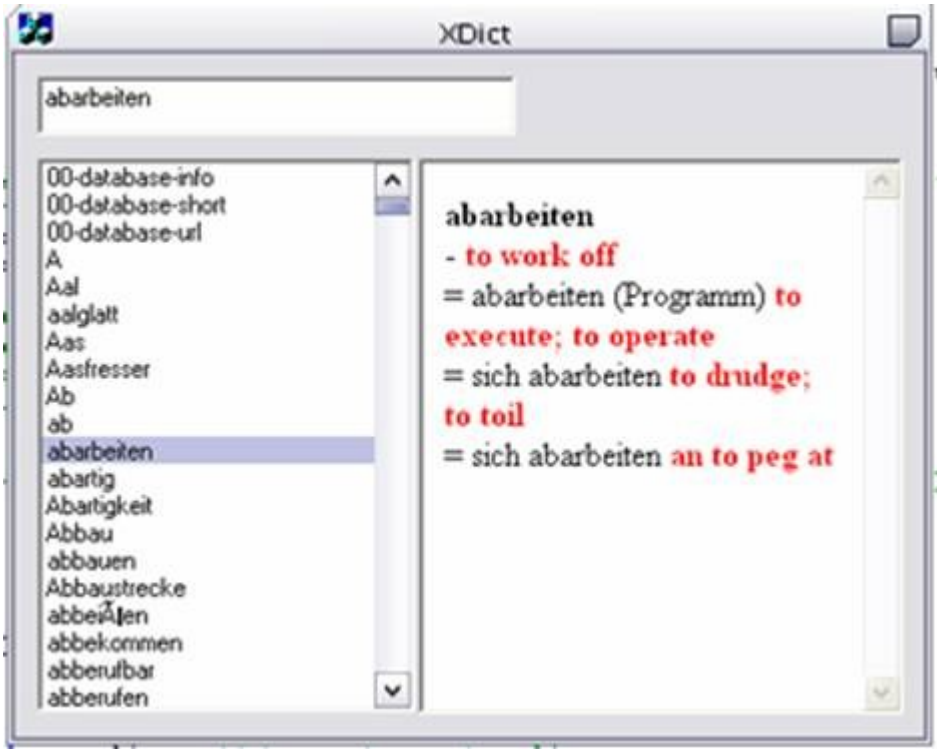
```
	BOOL gm = GetMeaning();
	if (gm)
	{
		m_wordlist.GetText(m_wordlist.GetCurSel(),m_word);
		UpdateData(FALSE);
		// Lay duong dan cua thu muc hien thoi
		DWORD cchCurDir;
		LPTSTR lpszCurDir;
		TCHAR buffer[MAX_PATH];
		lpszCurDir = buffer;
		GetCurrentDirectory(cchCurDir, lpszCurDir);
		CString str = lpszCurDir;
		str = "file://"+str+"\\temp.htm";

		//Hien thi nghia cua tu
		m_wordmean.Navigate(str,NULL,NULL,NULL,NULL);
	}
	else
	{
```

```
        MessageBox("Can't get the meaning of the word");
    }
```

- Tìm kiếm từ trong list box tương ứng với sự thay đổi trong từ nhập trong điều khiển Edit. Bạn cài đặt phần mã này vào sự kiến EN_CHANGE của điều khiển Edit. Như vậy mỗi khi bạn đánh 1 ký từ vào điều khiển Edit, chương trình sẽ tự động tìm một từ gần giống nhất trong danh sách các từ.

```
    UpdateData(TRUE);
    m_wordlist.SelectString(-1,m_word);
```

Sau khi xây dựng thành công, chương trình khi chạy sẽ có giao diện như sau



Như vậy trên đây, tôi đã trình bầy giúp bạn cách thức xây dựng một ứng dụng từ điển đơn giản. Với những sáng tạo và các kiến thức đã có, bạn hoàn toàn có thể bổ xung những khả năng mới cho ứng dụng từ điển này để nó không thua kém gì so với Click and See hay MTD. Ngoài ra, với một format từ điển khá đơn giản như thế này, bạn có thể dễ dàng xây dựng thêm một ứng dụng nữa để tạo từ điển mới một cách tự động. Tôi cũng xin giới thiệu cho bạn một số địa chỉ các phần mềm có sử dụng các bộ từ điển theo chuẩn của www.dict.org để các bạn tham khảo thêm:

- Trang Web của bạn Hồ Ngọc Đức. Địa chỉ: http://www.informatik.uni-leipzig.de/~duc/Dict/ . Tại đây bạn có thể tra từ điển trực tuyến. Hoặc download ứng dụng viết bằng java và các file từ điển về máy để chạy trực tiếp trên máy.

- PowerClick:. Địa chỉ:http://www.ttdomain.net/ttdownload. Phần mềm được giới thiệu trên PCWorldVN 7-2004, có khả năng tra theo kiểu Click And See trên một số ứng dụng

- E-lexikon: Địa chỉ: http://www.edusoft.com.vn. Phần mềm được giới thiệu trên PCWorldVN 7-2004, chạy theo mô hình Client - Server.

- MultiDictionary: Địa chỉ: http://huybien.vze.com/. Ứng dùng đa từ điển, giao diện đẹp, có khả năng phát âm các thức tiếng Anh, Nga, Pháp, Đức.

Chúc các bạn thành công và hài lòng với ứng dụng từ điển của mình.

Trần Bình An

Tự Động Hoá 3 - K46 Đại Học Bách Khoa Hà Nội.

Cơ chế load danh sách nhanh

Theo một cách thông thường như hướng dẫn của chuẩn dict, để load danh sách cách thông thường của chúng ta là nạp toàn bộ danh sách từ vào listbox, với cách này, các thao tác với danh sách từ sẽ rất đơn giản vì listbox đã hỗ trợ hết, nhưng nếu số lượng từ của từ điển tương đối nhiều một chút thì sao nhỉ, dẫn chứng nhé:
Từ điển echip, đây là trang chủ:
http://www.echip.com.vn/echiproot/html/tudienechip/quydinhsudung.html
còn đây là từ điển có dữ liệu lớn nhất của họ (từ điển hàn việt hơn 60.000 từ):
http://www.echip.com.vn/echiproot/html/tudienechip/echip_han_viet.rar
Hãy bật nó lên và xem thời gian nó khởi động , chúng ta khó ước tính được thời gian load danh sách vì thời gian đó gộp chung vào thời gian khởi động chương trình, vậy hãy bấm nút đổi từ điển và chọn chính từ điển hàn việt đang chạy, từ điển echip sẽ nạp list lại từ đầu, và chúng ta có thể ước lượng được thời gian load danh sách (như máy của mình là 9 giây).
Tiếp đến là multidictionary của anh huy biên, cái này có cải tiến, tốc độ load danh sách đã tăng lên đáng kể (khoảng gấp đôi), nhưng cũng không thoả mãn về tốc độ cần đạt được, load cái từ điển cỡ prodic 400000 từ thì cũng mất cả phút như chơi, mà load toàn lên ram, máy nặng lắm , đó là lý do vì sao mỗi khi chuyển tab thì cái multidictionary lại hiện cái hình mất vài giây trước khi chạy được.
Nếu những soft từ điển miễn phí đều như thế thì khó lòng chúng ta có thể thay thế các soft thương mại được.

Nhưng mình đã tìm ra được cách có thể load từ rất nhanh đối với bất kể dữ liệu từ điển nào dù lớn đến đâu chăng nữa. Phương pháp là load vài từ đủ để xem thôi, điển hình như stardict, load danh sách ngắn tí vài chục từ.
Cách này có vẻ giải quyết được phần nào nhưng vẫn chưa phải là hay, vì sao lạc việt, prodic , englist study vẫn có thể dùng cái thanh cuốn kéo từ đầu đến cuối hàng chục thậm chí hàng trăm nghìn từ được mà khởi động vẫn nhanh.
Mình đã tìm ra cách giải quyết vấn đề này, mình vẫn load mấy chục từ thôi(vừa đủ để list box không hiện ra thanh cuốn của nó). Thay vào đó mình lấy một cái VScrollbar lắp vào bên cạnh.
Vậy lắp vào làm gì (dở hơi chăng) , không phải đâu, nếu thế thì mình còn viết tut này làm gì.
vscrollbar có 3 thuộc tính bạn cần quan tâm là: Minimum, maximum và value
Minimum hãy đặt là 1, maximum = tổng số từ của từ điển đó , còn value , chính là để thể hiện vị trí tương đối của cái con trượt trên thanh cuốn đó.
Khi người dùng cuốn danh sách bằng vscrollbar thì value sẽ thay đổi, ví dụ value =20.000 (từ thứ 20 nghìn),ta sẽ nhảy đến vị trí từ thứ 20000 này , thay mấy chục phần tử listbox cũ thành các từ từ vị trí 20.000 -> 20.020 chẳng hạn.
Người dùng sẽ có cảm giác y hệt như cách khởi động lâu kia mà tốc độ khởi động của chúng ta thì nhanh vô cùng.
Chú ý:

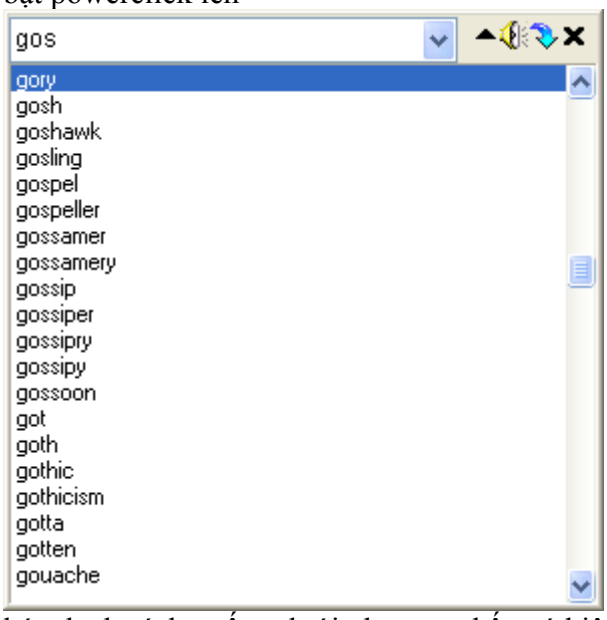
- Để sử dụng được kỹ thuật này , định dạng từ điển phải có khả năng truy xuất nhanh đến một vị trí bất kỳ trong từ điển
- Khi load kiểu này nếu bắt sự kiện cuốn danh sách, có thể từ điển không load kịp so với động tác cuốn của người dùng , có nhiều cách xử lý , spdict dùng 1 timer cứ 50 mili giây sẽ kt vị trí cũ có thay đổi so với giá trị của vscrollbar không (dùng 1 biến để lưu vị trí cũ) , làm vậy thì tối đa 50 mili giây mới phải load danh sách, danh sách sẽ đủ thời gian thực hiện load , 50 mili giây là khoảng thời gian rất nhỏ , người dùng sẽ không thể phân biệt được
- khi resize form thì phải tính toán số từ cần hiển thị trên listbox nhé, công thức đây này:
  số phần tử = (độ cao listbox-4)/độ cao một phần tử { -4 là trừ đi viền trên và dưới listbox đó}
- Do không phải là listbox thật, để làm nó giống , các bạn cần xử lý khi người dùng ấn Home / end (nhảy về đầu , cuối danh sách) page up , page down , cuộn chuột giữa , dùng phím mũi tên di chuyển trên dưới ... (tham khảo code spdict nhé)

Không phải chỉ mỗi mình mình mà nhiều từ điển khác cũng làm cách tương tự:

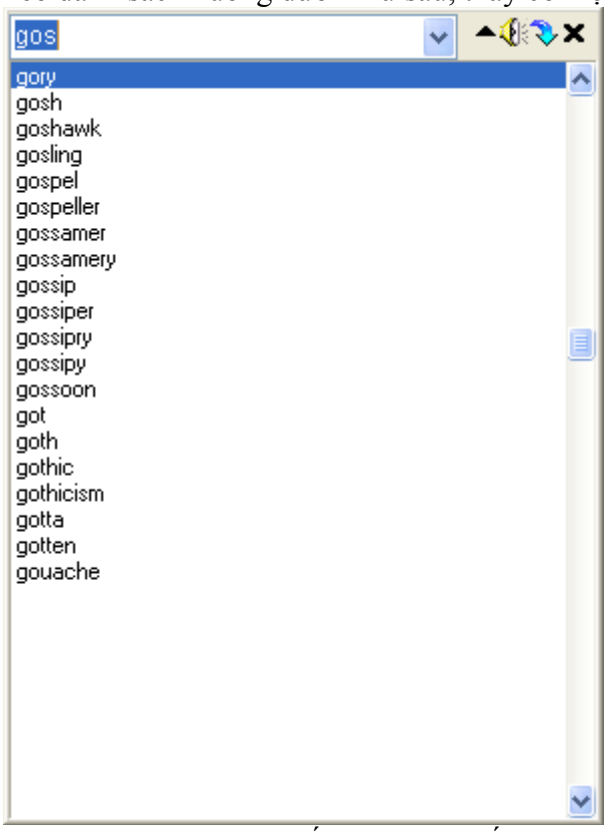
đây là bài viết của mình bên vnoss.org sau khi dùng thử vdict của họ tính đến hiện nay ,mình thấy có powerclick(load vừa đủ) , vdict load 100 từ và jtranslator load 50 từ:

> hix, tưởng thế nào, cái này load có 100 từ à , chắc dùng cái vscrollbar đè thẳng lên cái scrollbar của listbox nên khi thay đổi kích thước form không bị giật như tui . Nói chung là cùng tư tưởng giống nhau.
> Thực ra chưa đọc được code, nhưng có xem qua, nhưng dùng chuột giữa di chuyển thấy được một đoạn

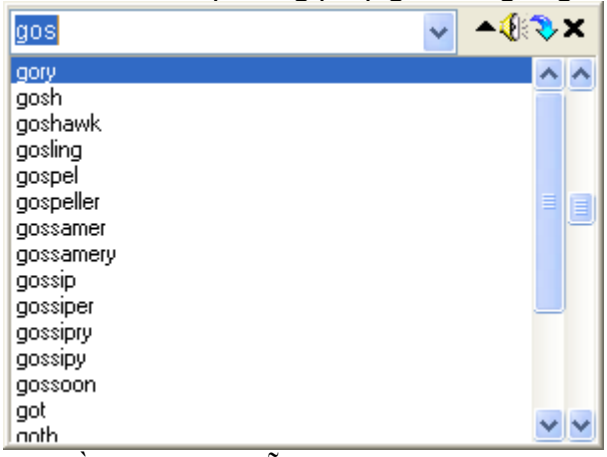
rồi tắt (đếm thấy có 100 phần tử). Thực ra cũng bằng thủ thuật tui cũng phát hiện ra powerclick cũng làm cách tương tự :
bật powerclick lên



kéo danh sách xuống dưới như sau, thấy có hiện tượng lạ, danh sách bị đơ rồi



Không phải đâu, bản chất của nó là thế này này này, nó load nhanh là do cũng dùng scrollbar giả lắp vào và load với một phương pháp gần như giống hoàn toàn tui (tư tưởng lớn gặp nhau).



Lúc đầu tui tưởng mỗi mình thông minh (^_^) , mấy soft thương mại làm được vì nó có tiền, không thèm chấp. Hoá ra về sau phát hiện ra mấy soft này thì hoá ra cùng cơ chế, đã thế powerclick và vdict , cả

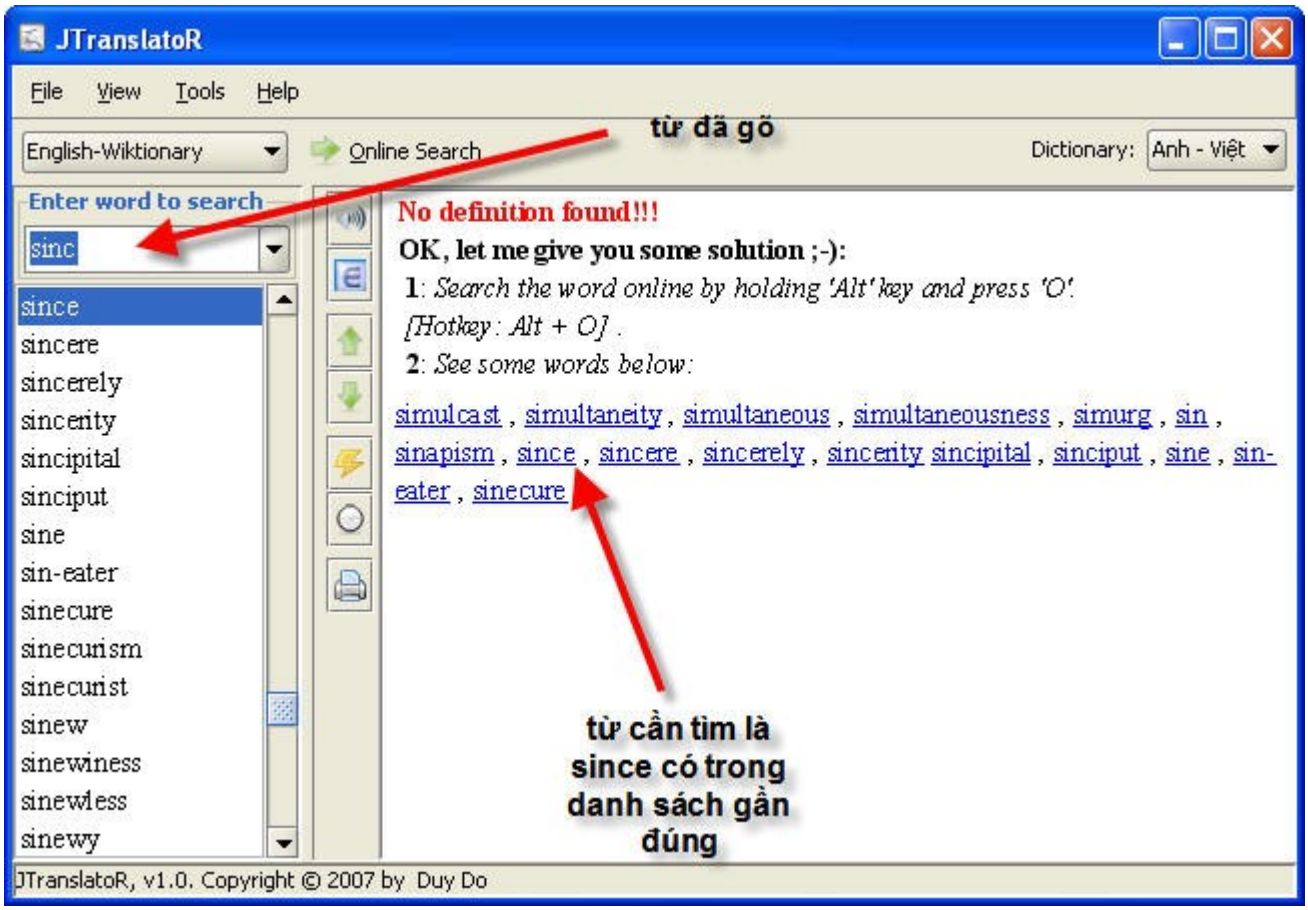
spdict đều phát triển từ chuẩn dict (bài viết của anh peacemoon trên quản trị mạng).

@To bé wasabi : http://forums.congdongcviet.com/showthread.php?t=3186
ở mấy bài viết đó cũng toàn thành viên lão làng bên cviet nhận xét đó, csdl tuy mạnh nhưng các soft thương mại chả cái nào dùng đâu, đơn giản vì nó đa năng nhưng nó không thể bằng một cái chuyên dụng được , với lại khó quản lý bao quát hết cũng như bị lệ thuộc . Cũng giống như máy tính có thể nghe nhạc, xem ti vi, nhưng người ta vẫn sắm riêng lẻ những thứ đó.

Tìm kiếm nâng cao cho từ điển

Ba phần, ráng đọc cho hết :

1 .Làm cái từ điển cho ra trò thì cũng phải có vài tính năng ngon ngon một chút , giờ đến tìm kiếm nâng cao , như check spelling của babylon hay fuzzy(truy vấn mờ) của stardict , nhưng chúng ta sẽ làm những kỹ thuật thật đơn giản nhưng hiệu quả phải cao , sau nhiều ngày tìm kiếm cuối cùng mình đã tìm thấy cái này ở JTranstator . Lúc đầu mình sửng sốt , sao nó làm siêu thế mà tốc độ thì nhanh kinh (gần như ngay lập tức - đến babylon cũng phải mất mấy giây)



Hoá ra kỹ thuật là thế này, nó đơn giản load 7 từ trước danh sách và 7 từ sau danh sách một từ ở vị trí gần từ gõ nhất (nhìn sang danh sách bên trái thì sẽ thấy đó là since.
Thủ thuật chỉ có vậy, rất nhanh phải không , nhưng cũng rất có ích đó, ví dụ các thì quá khứ thêm ed hay ing hay ... thì số trường hợp nằm trong danh sách gần đúng đó rất nhiều . Mà nguyên lý thì quá đơn giản. Tuy đơn giản nhưng hiệu quả đó , không phải chỉ là giả tạo cho giống babylon đâu.

Nhưng spdict là từ điển đa ngôn ngữ , tra nhiều từ điển , nếu bê nguyên thì nó chỉ tìm thông minh được cho 1 từ điển (như ở trên là anh việt) , nhỡ các từ ở từ điển việt anh lại gần đúng hơn thì sao , vậy chúng ta sẽ áp dụng thủ thuật trên cho tất cả các từ điển , rồi trộn chúng lại như sau :

1. Cho chúng vào 1 mảng có kích thước 15 * số từ điển , bằng cách lấy mỗi từ điển 15 từ gần đúng rồi tống vào
2. Các từ đó sắp xếp theo các mã sắp xếp khác nhau, phải soft lại theo mã en-US (mã chung nhất)
3. Lọc các phần từ trùng nhau (ví dụ từ điển việt anh và việt việt là lắm từ trùng lắm)
4. tìm vị trí của từ cần tra trong danh sách từ đã lọc
5. Hiển thị 7 từ trước, từ gần nhất và 7 từ sau giống như trên

Cách làm này còn để áp dụng cho việc load danh sách chung giống các từ điển stardict hay babylon (dùng cho spdict small từ bản 4 trở lên)

2 . Tìm kiếm với wildcard và regular expression:

Nếu chỉ có như trên thì ít quá , không thể gọi là advance search được , spdict còn hỗ trợ wildcard và regular expression . Java chỉ có regular expression thôi , tuy nhiên regular expression là mở rộng của wild card (nhưng wildcard đơn giản với người dùng hơn nên vẫn cần phải sử dụng chứ không bỏ được) , đây là hàm convert chuỗi regex sang wildCard (dùng regex mà trả về kết quả như wildCard) :

```
    private static String replaceWildcards(String wild) {
    StringBuffer buffer = new StringBuffer();
    char[] chars = wild.toCharArray();
    for (int i = 0; i < chars.length; ++i) {
      if (chars[i] == '*') {
        buffer.append(".*");
      } else if (chars[i] == '?') {
        buffer.append(".");
      } else if ("+()^$.{}[]|\\".indexOf(chars[i]) != -1) {
        buffer.append('\\').append(chars[i]);
      } else {
        buffer.append(chars[i]);
      }
    }
    return buffer.toString();
  }
```

Cách thức dùng regex thì tự search

3. Tìm từ gần đúng:

Cái này mình tự chế , tư tưởng như sau:

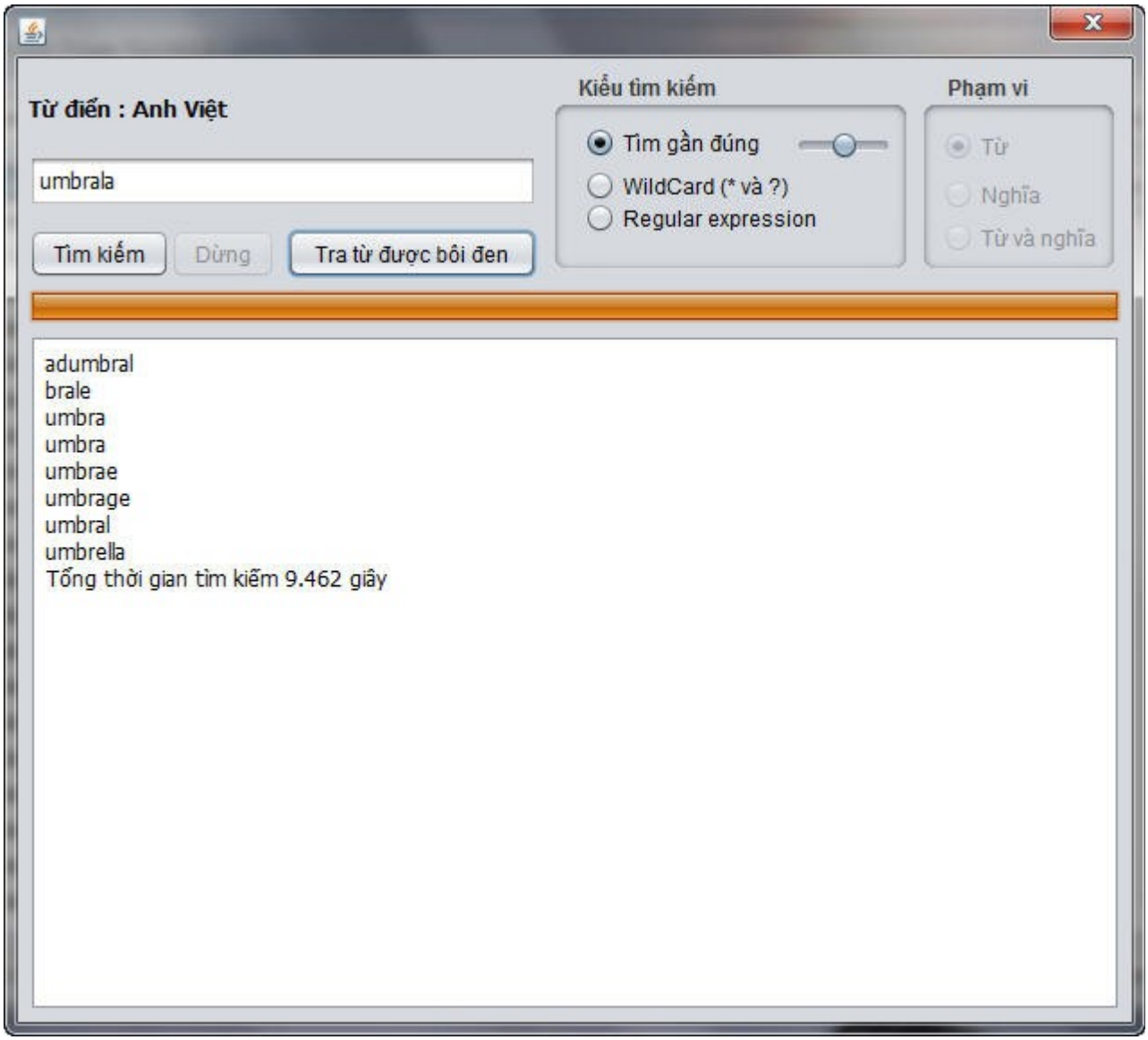
+Đầu tiên kiểm tra độ dài string so sánh :ít hay hơn 30% thì loại (có thể tăng giảm cái này để tăng hoặc giảm số từ tìm được)
+Tiếp là so sánh từng ký tự của 2 string nếu không bằng nhau thì lỗi cộng thêm 1, so sánh các từ lân cận tiếp theo của cả 2 string ,
Trong khoảng sai số nếu có , thì chỉnh lại vị trí i,j là chỉ số của 2 string đó, cái này sẽ kiểm tra các lỗi thừa hay thiếu từ , đọc ký tự kế tiếp .
+Cuối cùng , khi 1 trong 2 string đã đi hết
thì còn mẩu đuôi ta làm : loi += s.Length - i + s1.Length - j;
tức là nếu 1 string còn thừa thì cho mẩu đó là lỗi cộng vào
nếu số lỗi <=30% thì là đạt , không thì không đạt , đơn giản vậy thôi

code này :

```
    public class ApproximatString {
      String s;
      int i, j, k, loi, saiSo;

      public ApproximatString(String nhap, float phantram) {
        s = nhap;
```

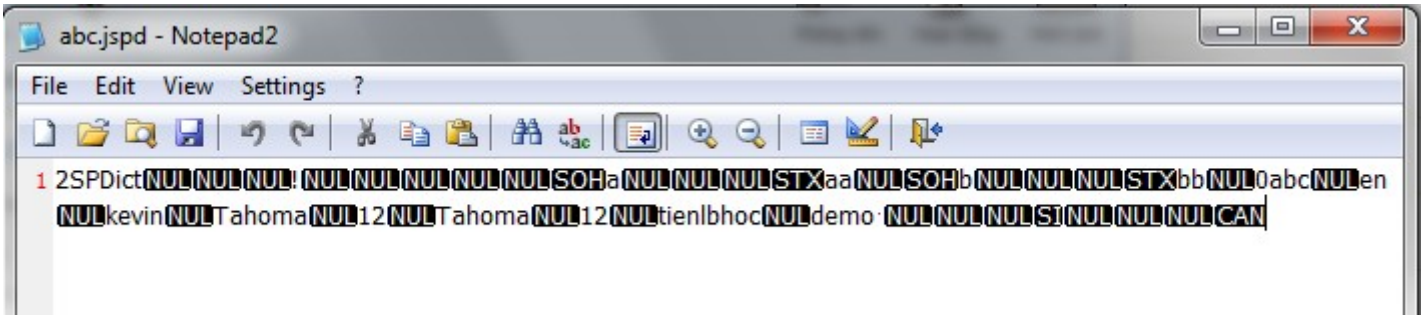
```
        saiSo = (int) Math.round(s.length() * phantram);
    }
    public boolean SoSanh(String s1) {
        if (s1.length() < (s.length() - saiSo) || s1.length() > (s.length() + saiSo)) {
            return false;
        }
        i = j = loi = 0;
        while (i < s.length() && j < s1.length()) {
            if (s.charAt(i) != s1.charAt(j)) {
                loi++;
                for (k = 1; k <= saiSo; k++) {
                    if ((i + k < s.length()) && s.charAt(i + k) == s1.charAt(j)) {
                        i += k;
                        break;
                    } else if ((j + k < s1.length()) && s.charAt(i) == s1.charAt(j + k)) {
                        j += k;
                        break;
                    }
                }
            }
            i++;
            j++;
        }
        loi += s.length() - i + s1.length() - j;
        if (loi <= saiSo) {
            return true;
        } else {
            return false;
        }
    }
}
```

Còn đây là kết quả :

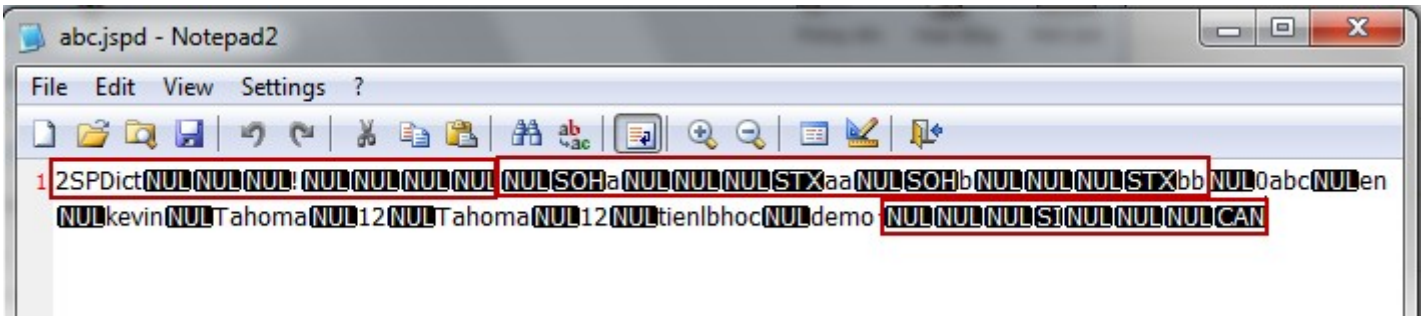


Định dạng từ điển SPDict

Mình đã thay đổi định dạng vài lần rồi , bản 6.0 lại vừa thay đổi lần nữa cho nó tốt hơn , giờ đã tìm được một định dạng từ điển giải quyết được khá tốt các vấn đề , mà định dạng cũng rất dễ (pro đọc 10 phút hiểu hết) . Đây là ví dụ 1 file từ điển abc gồm 2 từ :a->aa và b->bb được mở bằng notepad 2, các bạn có thể dễ dành nhận ra vị trí của dữ liệu 2 từ này trên hìn



Có thể phân định dạng spdict ra làm 4 phần (3 phần đóng khung đỏ và 1 phần không đóng khung được vì xuống dòng như hình vẽ :



* Phần thứ nhất gồm chuỗi 2SPDict ở đầu file (để đánh dấu file này là của từ điển spdict tạo ra mà không phải là 1 file nào đó đổi đuôi thành.

4 byte tiếp theo mà bạn thấy là null null null ! ấy , nó lưu vị trí của phần thứ 3 (phần không đánh dấu đỏ)

4byte tiếp lưu số dữ liệu thừa phát sinh trong quá trình làm từ điển (hiện mới tạo nên nó = 0 , 4 chữ null)

* Phần thứ 2 chứa dữ liệu của từ điển :

Bạn sẽ thấy null sqh rồi mới đến a (từ đầu tiên) , đây là 2 byte dạng short lưu độ dài của từ (có giá trị 1)

sau đó là nghĩa của từ a , lưu độ dài bằng 4byte (null null null stx) rồi đến aa . Tiếp theo là từ b → bb , bạn sẽ thấy các byte lưu độ dài string có độ dài giống hệt từ đầu tiên a → aa.

* Phần thứ 3 :nó có giá trị tương tự như 1 nội dung từ ở phần 2 (2 byte lưu độ dài) , phần còn lại là nội dung là một chuỗi gồm nhiều chuỗi con phân cách nhau bởi byte có giá trị 0 (null) gồm :

- Tên từ điển (abc)
- mã sắp xếp(en)
- giọng phát âm(kevin)
- font , kích thước từ và nghĩa(tahoma , 12, tahoma ,12)
- tác giả (tienlbhoc)
- thông tin thêm (demo)

* Phần thứ 4 : gồm 8 byte , là 2 số integer (tương ứng với 2 từ) , mỗi số lưu vị trí của một từ (a và b trong phần thứ 2 của từ điển) . Có thể nói đây là danh sách vị trí hay gọi là con trỏ văn bản (đã được sắp xếp) , thuật toán tìm kiếm sẽ được diễn ra ở đây

==========

Cách đọc file từ điển :

Đầu tiên ta đọc 7 ký tự đầu xem có phải là 2SPDict không , đọc 4 byte tiếp để được số interger vị trí của thông tin từ điển và 4 byte nữa để lấy số dữ liệu thừa . Sau đó nhảy đến vị trí của phần thứ 3(thông tin) , đọc 2 byte để lấy độ dài mảng byte chứa thông tin từ , đọc mảng byte chứa thông tin từ → convert ra chuỗi ký tự , split các string thành phần phân cách bởi ký tự '\0' ta được các thông tin của từ điển .

Đọc hết thông tin phần thứ 3 thì con trỏ file sẽ ở vị trí của phần thứ 4 , lưu vào và ta bắt đầu tìm kiếm.

Đơn giản vậy thôi à , thế thì hay ở chỗ nào. đầu tiên để lấy được tổng số từ ta lấy kích thước file - vị trí phần thứ 4 rồi tất cả chia cho 4 (vì mỗi vị trí từ dùng 4 byte ở phần thứ 4 để lưu nên lấy kích thước nó chia cho 4 sẽ ra số byte). để nhảy đến vị trí thứ n bất kỳ thì ta chỉ việc nhảy đến vị trí mảng +4*(n-1) .
Vì đây là mảng đã sắp xếp (sắp xếp trong quá trình convert dữ liệu và thêm xóa), việc tìm kiếm nhị phân có lẽ không phải nói nhiều chắc ai cũng biết chỉ có điều mảng bình thường thì ta gõ chỉ số nó ra biến có nội dung ngay ví dụ a[3] , nhưng vì đây là mảng ta tự định nghĩa nên phải nhảy đến phần tử cần lấy nội dung , đọc 4 byte lấy vị trí nội dung , nhảy đến từ rồi mới lấy được nghĩa .

Để thêm xoá từ, ví dụ thêm từ ở vị trí thứ n (để cho danh sách vẫn theo thứ tự abc) , ta tống toàn bộ mảng con trỏ (gọi

thế này cho quen thuộc) lên ram , ghi nội dung từ mới vào cuối phần dữ liệu từ , lưu phần mảng từ đầu -> n-1 vào file, thêm vị trí phần tử mới thêm , lưu nốt phần còn lại vào , setlength lại cho file . Phần này giảng hơi chuối, nếu các bạn không hiểu thì có để đọc code.
Xoá từ, chúng ta chép đè phần từ và nghĩa thành các byte =0 hết (để khi tìm kiếm nâng cao , phần này sẽ không có trong kết quả, chỉ xoá phần vị trí của từ cần xoá trong mảng con trỏ . Giống access vậy, dữ liệu này sẽ là dữ liệu thừa , và chỉ được đẩy đi khi dùng lệnh "compact and repair database". Chúng ta phải tạo thêm 1 công cụ có chức năng tương tự cho từ điển.
Sửa và đổi tên từ các bạn tự code theo phương hướng trên .

Đối với việc cập nhật thông tin từ điển (font , size , phát âm...) thì vì phần thông tin không đặt ở đầu từ điển nên khi cập nhật chỉ việc lưu phần danh sách vị trí lên ram , ghi đè thông tin mới lên thông tin cũ rồi ghi lại danh sách vị trí . Ở các phiên bản trước bản 6 thì dữ liệu được ghi ở đầu file , ví dụ từ điển 50 mb phải dịch cả 50 mb về phía sau rồi ghi lại thông tin từ , bản 6.1 trở lên thì chỉ phải dịch phần danh sách vị trí nên nhanh hơn rất nhiều

Vậy chắc sẽ có bạn hỏi, làm vậy phải load toàn bộ mảng hàng trăm nghìn từ (nếu từ điển lớn) lên thì chết à . Mình đáp rằng hãy làm một con tính 4byte * 100.000 chưa được 400KB , hãy tưởng tượng copy 400KB nhanh thế nào thì tốc độ thêm xoá từ cũng nhanh như vậy.

Cần lưu ý phần mảng con trỏ phải luôn ở cuối file để kích thước của nó luôn luôn là min (4byte cho mỗi phần tử) , chứ nếu để mảng này ở đầu file, load lên ram , phải load luôn cả phần nội dung đằng s (có khi hàng trăm mb) thì chết mất .

Nói chung so với các loại cây thì cái này thêm xoá chậm hơn nhưng đối với từ điển , nhập tuần tự chứ không thay đổi liên tục thì tốc độ này hoàn toàn thoả mãn. Người dùng sẽ không thể nhận thấy sự khác biệt nếu thời gian nhập từ là 0.001 giây với 0.1 giây . Định dạng này rất thích hợp khi thêm xoá với từ điển. Mọi người cũng có thể ứng dụng cái để làm các csdl loại nhỏ, cơ động , không đòi hỏi quá nhiều.

Chú ý:

Đối với tra từ nâng cao , không nên áp dụng nhảy đến phần tử thứ n như trên cho toàn bộ từ điển (phải mất công nhảy vị trí liên tục trong file gây tốc độ chậm) , cần để ý rằng từ điển gồm 4 phần , phần 2 là các từ và nghĩa nối tiếp nhau , ta chỉ việc đọc các từ nối tiếp nhau cho đến hết phần này thì thôi

Do lưu bằng 4byte cho mỗi phần tử , nên giới hạn dữ liệu từ điển là 2Gb , vì 4byte chỉ chứa được vị trí đến thế , nói chung là thừa thãi, từ điển thường không bao giờ hết (từ điển thường chỉ cỡ vài mb hay vài chục mb thôi) , nhưng cứ nói thế cho đầy đủ

Để đọc và thao tác định dạng này bạn chủ yếu cần chú ý 2 đoạn code sau :

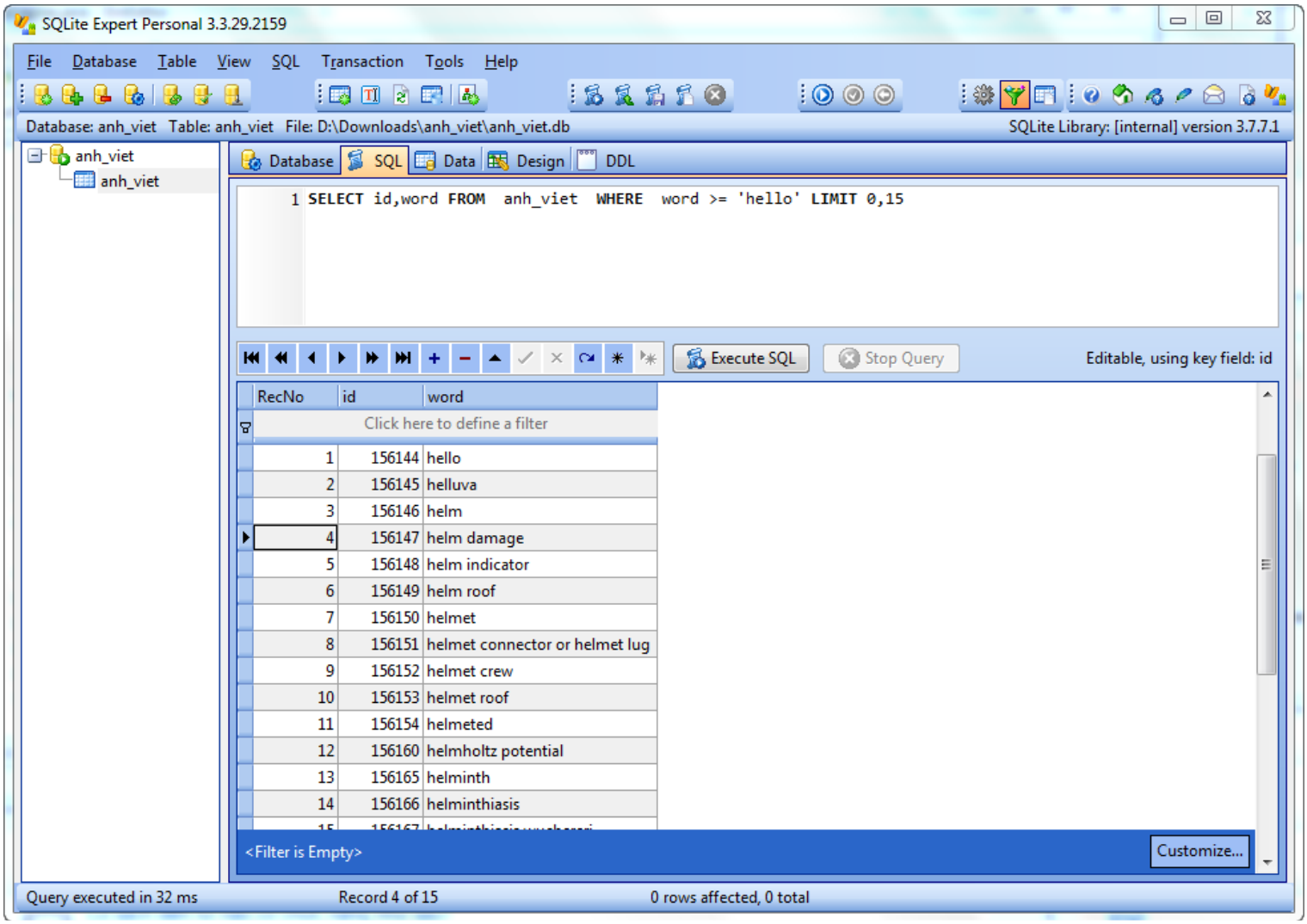
Nhảy đến vị trí bất kỳ trong file bằng hàm seek

```
RandomAccessFile raf = new RandomAccessFile(s, "rw");

raf.seek(1000); //nhảy đến vị trí 1000 byte
```

Lấy kích thước số byte của 1 string , convert từ byte sang string

```
import java.io.*;

class CLab
{
	public static void Main()
	{
		String s = "tiến";
		byte[] b=s.getBytes("UTF-8");  // chuyển sang mảng byte
		tu = new String(bs, 0, doDaiString, "UTF-8");  // chuyển mảng byte sang string
	}
}
```

Giới thiệu luôn các bạn ví dụ về dùng sqlite về cách truy xuất tìm kiếm, load danh sách , cái này là cơ sở dữ liệu anh việt của andict (android) gần 400.000 từ , đoạn truy vấn này sẽ load 15 từ ngay sau chữ hello.



Những giải pháp chưa hoàn thành

Những tính năng chưa hoàn thành

Mình trình độ vẫn còn kém, dù đã cố gắng tìm kiếm nhiều biện pháp khắc phục nhưng vẫn còn mấy tính năng chưa hoàn thiện được:

+Click and see : nghe nói đây là một kỹ thuật cực khó , các soft trong nước hiện nay chưa cái nào có thể bắt text gọi là tốt cả , nhất là pdf , hiện nay các thư viện lập trình sẵn trên mạng chỉ có cái wordcaptureX ngoại trừ không bắt được trong openoffice ra thì nó bắt text gần như ngang ngửa babylon , các bạn có thể search trên mạng , có thể down demo của nó về dùng thử, ai định mua (chưa có crack đâu) thì giá 900USD đó .Hoặc một cái khác miễn phí , cũng ổn là cái bắt text của stardict , nhưng mình không biết dịch mã nguồn nó thế nào ,nêu hiện tại SPDict chỉ dùng tra từ qua clipboard, chỉ bắt được text có khả năng bôi đen, menu , lable thì chịu
+Tìm kiếm nâng cao(Kiểm tra chính tả – tìm gần đúng) : dù sao cũng chỉ là code mình tự chế , không thể pro được bằng các bộ chuyên nghiệp được, mình cũng nghĩ có mấy ngày thôi , mình nghĩ có thể phát triển hơn.
+Văn bản multimedia : webbrowser để tăng lực đã phải lược đi những tính năng cơ bản film, nhạc, flash, scrip... chỉ dùng được ảnh thôi .
+Nếu cải tiến được thêm để từ điển không bị phát sinh dữ liệu thừa thì tốt hơn

Liên kết online offline

Như mình đã nói ở bài giới thiệu , từ điển online có những nhược điểm:

+Không phải ai cũng lên net để tra và không phải ai cũng có net , chưa kể nhiều người xót tiền net nữa

+Đôi khi người dùng chỉ muốn tự tạo riêng cho mình để ghi chép vấn đề gì đó hay chuyên ngành gì đó cho riêng mình hay chia sẻ nội bộ . Từ điển online hiện giờ không thể làm được điều đó
+Vì là web nên ví dụ các tính năng kiểu như click and see , load đầy đủ danh sách ... không thể có
+Web có thể dừng hoạt động bất cứ lúc nào
+Dữ liệu thêm vào trang web sẽ không lấy lại được, như vậy , phải gắn bó với nó mãi mãi , giống như google không thể chat với yahoo , khi người dùng không thích dùng web này , thì khi sang web khác, có thể sẽ phải nhập lại các từ trước kia .

Nhưng nó cũng có những ưu điểm mà từ điển offline không có :

+Không cần cài đặt
+Cập nhật thường xuyên
+Nếu máy bị sự cố thì vẫn có dữ liệu lưu trên server
+Có thể hỏi đáp , tranh luận về một từ nào đó còn thắc mắc
....

Nếu chúng ta kết hợp được cả 2 loại thì đúng là rất tốt :
Ví dụ từ cập nhật vào từ điển , sau khi tích đến một số lượng nào đó phần mềm sẽ đề nghị gửi dữ liệu đó lên web, để ban biên tập tổng hợp , cập nhật . Khi máy bị lỗi , ta có thể down lại dữ liệu đã up + với dữ liệu của nhiều người khác cập nhật thì quả là quá hay , ta bỏ công nhập 10 từ , có thể sẽ thu lại cả nghìn từ (nếu có 100 người cùng làm như bạn) , chưa kể dữ liệu đó còn được biên soạn cho hợp lý , sửa lỗi ... . Đó quả là một mô hình phát triển rất tốt. Mình thấy các trình antivirus hay window bị lỗi đều yêu cầu gửi online để xử lý nên việc áp dụng cho từ điển cũng chẳng có gì là không đúng . Hix đáng tiếc , mình không có chút kiến thức nào về web nên không thể giúp gì được trong vấn đề này , nhưng mình viết bài này để người phát triển kế thừa dự án có thể làm tiếp những thứ mình chưa làm .