[Mục Lục 1](#_Toc498704643)

[**I.** **CÁC MODULE CỦA PROJECT** 2](#_Toc498704644)

[**II.** **SƠ ĐỒ HOẠT ĐỘNG CHÍNH CỦA MODULE GÕ** 5](#_Toc498704645)

[**III.** **CÁCH ĐỊNH NGHĨA 1 LOẠI KEY CATEGORY MỚI:** 8](#_Toc498704646)

[**IV.** **ĐỊNH NGHĨA BẢNG MÃ CỦA TIẾNG MƯỜNG** 10](#_Toc498704647)

[**V.** **ĐỊNH NGHĨA KIỂU GÕ TELEX MƯỜNG** 12](#_Toc498704648)

[**VI.** **THÊM TÍNH NĂNG CHỈNH SỬA SAU KHI GÕ PHÍM** 16](#_Toc498704649)

[**VII.** **THÊM PHÍM TẮT CHO BỘ GÕ MƯỜNG** 19](#_Toc498704650)

1. **CÁC MODULE CỦA PROJECT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byteio** | **Keyhook** | **Newkey** | **Uvconvert** | **Vnconv** |

**BYTEIO**

Quản lý và build những file Pre –compile header.

Dùng để chỉ ra các phần của source mà chúng ta thêm vào hoặc tùy chỉnh

**HOOK**

Hook là một cơ chế dùng để chặn và lắng nghe các sự kiện, như sự kiện click chuột, gõ phím, đồng thời xử lý các công việc khi gặp sự kiện đấy. Trong opensource của Unikey, Hook được định nghĩa ở trong thư viện keyhook

* **File keyhook.cpp**

Trong file này chỉ cần quan tâm đến 1 số hàm chính:

* Hàm MyKeyHook () được gọi liên tục, lúc nào cũng lắng nghe sự kiện phím và đưa ra những xử lý.
* Hàm PushBuffer() đây là hàm chỉ được gọi khi cần dùng đến buffer, những trường hợp cần dùng đến buffer là các kí tự đặc biệt và dấu, đặc biệt là các kí tự cần dùng đến tổ hợp phím, ví dụ á (as) ư (uw).

Nguyên tắc xử lý buffer của hàm này sẽ là chuyển ký tự vừa gõ vào buffer, rồi in ký tự đó ra.

* Hàm ReleaseControlKey() Giải phóng các Phím control : như ALT, CTRL, SHIFT .vv để tránh trường hợp không phân biệt được chữ hoa với thường khi gõ phím, các tổ hợp phím.
* Hàm CheckBack() dùng để đảm bảo giải phóng ControlKey trước khi PushBuffer
* **File Vietkey.cpp**

Dùng để quy định các cách xử lý khi hook bắt được sự kiện ấn phím

Một số hàm cơ bản của vietkey.cpp

* Hàm PutChar() dùng với các kí tự đơn thuần k cần dùng tổ hợp phím, nó sẽ in thẳng các kí tự ra màn hình không cần kí
* Hàm process() là hàm xử lý các phím nhập vào, Hàm này sẽ kiểm tra xem phím mà Hook bắt được là phím nào, đóng vai trò là gì để đưa ra các loại xử lý tương ứng
* Hàm processBackspace() được gọi khi phím xóa backspace hoặc phím delete được gọi
* Hàm clearBuf() là hàm xóa buffer.
* Hàm ThrowBuf() là hàm giải phóng buffer khi tràn bộ nhớ
* Hàm keyCategory() hàm kiểm tra xem phím vừa nhập thuộc loại nào, phím Breve, hay Doublechar, hay phím dấu…
* Hàm putBreveMark() thả mũ – ngoặc cho các kí tự
* Hàm doubleChar() – xử lý các kí tự đơn thuần a,b,c.
* Hàm shortKey() xử lý các phím tắt []{}
* Hàm putToneMark() thả dấu – cho các kí tự

**NEWKEY**

Quản lý các giao diện của ứng dụng Unikey, vị trí các button, các text, và các chức năng, phụ trách việc mã hóa và quy định việc mã hóa.

* **File endcode.cpp**

File này quy định hầu hết mọi thứ, từ cách gõ TELEX, VNI cho đến các bảng mã được sử dụng, cách mã hóa tiếng việt.

* **File ukmenu.cpp**

File này quy định các item của 1 menu và các tính năng

**UVCONVERT**

Quản lý việc chuyển mã sang Unicode hoặc Unicode sang mã khác.

**VNCONVERT**

Quản lý bảng mã và việc chuyển đổi giữa các bảng mã

* **File data.cpp**

Định nghĩa các bảng mã

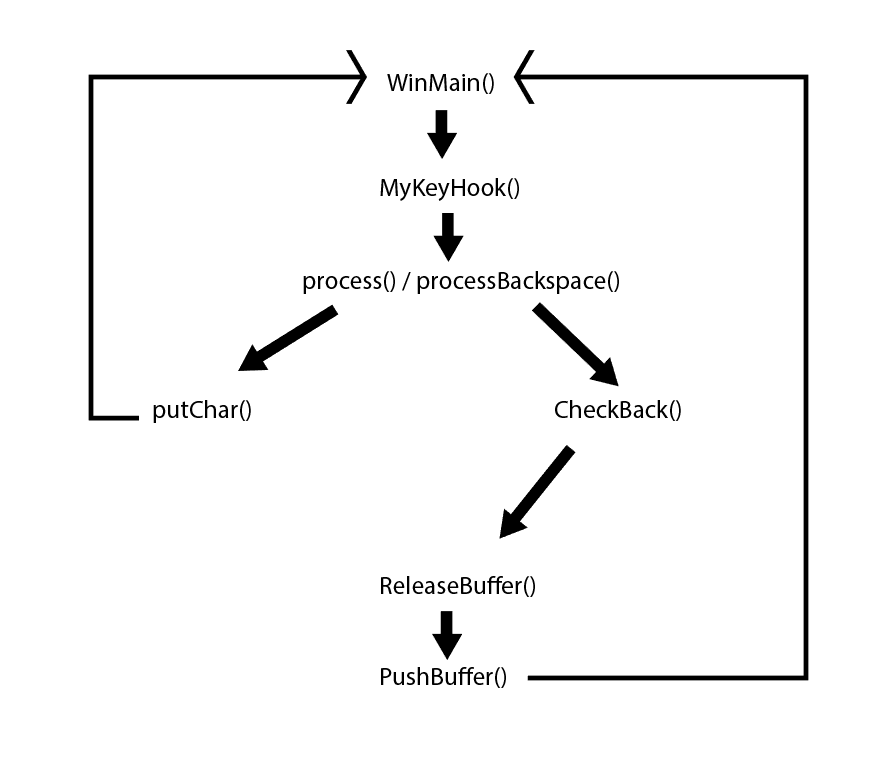
* **File charset.cpp**

Quản lý việc chuyển đổi các bảng mã, quản lý hành vi của từng bảng mã.

* **File convert.cpp**

Chưa rõ chức năng

1. **SƠ ĐỒ HOẠT ĐỘNG CHÍNH CỦA MODULE GÕ**



Cách thức hoạt động của hàm Process()

Hàm process luôn là hàm được gọi khi có sự kiện nhấn phím, các thao tác cơ bản của hàm process như sau.  
1. Kiểm tra xem phím vừa nhập là phím gì? Thuộc loại key category() nào ? Dưới đây là các loại của phím

1. BREVE MARK:

Là phím dùng để gõ dấu trăng, tức dấu móc của chữ ư, ơ, và ă. Theo như định nghĩa truyền thống của kiểu gõ Telex thì phím W đảm nhiệm vai trò này. Breve Mark là các phím có trong mảng TelexBreves[], được định nghĩa trong endcode.cpp

1. DOUBLE KEY:

Là các phím dùng để gõ các chữ đôi, ví dụ: Đ, Ê, Ô, Â…Các phím này được định nghĩa ở trong mảng DoubleChars[], trong endcode.cpp.

1. TONE MARK:

Là các phím dùng để gõ dấu câu, ví dụ: s, f, x, j, r… Các phím này được định nghĩa ở trong mảng TelexTones[] trong encode.cpp

1. SHORT KEY:

Các phím tắt, ví dụ [ ] { } w

1. VNI DOUBLE CHAR MARK

Các phím của VNI dùng để gõ chữ đôi Đ, Ê

1. ESCAPE KEY:

Chưa rõ

1. SEPARATOR KEY:

Các phím tạo khoảng cách, ví dụ, enter, tab…

1. SHORT SEPARATOR KEY:

Là phím space, nhưng bị xung đột với Seperator Key nên tác giả Phạm Kim Long đã tắt và comment nó đi.

2. Sau khi Xác định được key category là loại nào, dựa vào đó sẽ gọi hàm xử lý tương ứng.

Điểm khác biệt của Unikey Tiếng Việt và Unikey Mường

Vì bảng chữ cái tiếng Mường không có chữ Q ( cũng giống như chữ W trong tiếng việt), chính vì thế tôi đã tạo ra 1 loại Key Category mới, có chức năng nhân đôi nguyên âm trong từ. Tôi đặt tên nó là DUPLICATE MARK:. Khi phím Q được ấn, nó sẽ nhân đôi nguyên âm trong từ, nếu Q đứng đầu từ thì nó vẫn sẽ viết ra chữ Q bình thường.

1. **CÁCH ĐỊNH NGHĨA 1 LOẠI KEY CATEGORY MỚI:**

Trước khi tìm hiểu về Key category, tôi sẽ giới thiệu thêm về chuẩn mã hóa UTF 32. Mã hóa theo chuẩn UTF-32 tức là dùng 32 bit để lưu thông tin tiếng việt

Bit 0 – 4 : index of vowels

Vowels: là các nguyên âm

Bit 5 – 8 : index of macro key

Macro key là những phím tắt mà khi ấn nó sẽ thực hiện liền luôn 1 chuỗi các phím: Vd như

|  |  |
| --- | --- |
| W | Ư |
| W | Ư |
| [ | ư |
| ] | ơ |

Bit 9 – 13 : index of double characters:

Các kí tự trong bảng chữ cái tiếng việt cần ấn 2 lần để hiển thị: ư ơ

Bit 14 – 17 : tone mark:

Dấu câu:

Bit 18 – 21 determines the index of the tone associated with character

Dùng để xác định index của các dấu với 1 ký tự, Ví dụ, index của dấu sắc trong kí tự ‘á’ = 1

Bit 22 : Breve Mark

Bit này sẽ được đánh dấu nếu kí tự đó có dùng dấu mũ

Ví dụ dấu mũ trong chữ ă, chữ â , chữ ơ ấ bờ cờ dờ đờ e ê gờ mờ y

Bit 24 : Soft Separator

Phím space

Bit 25: Separator flag

Các phím enter, tab…

Bit 26 – 28: index of VNI double marks (‘6’, ‘7’, ‘8’, ‘9’)

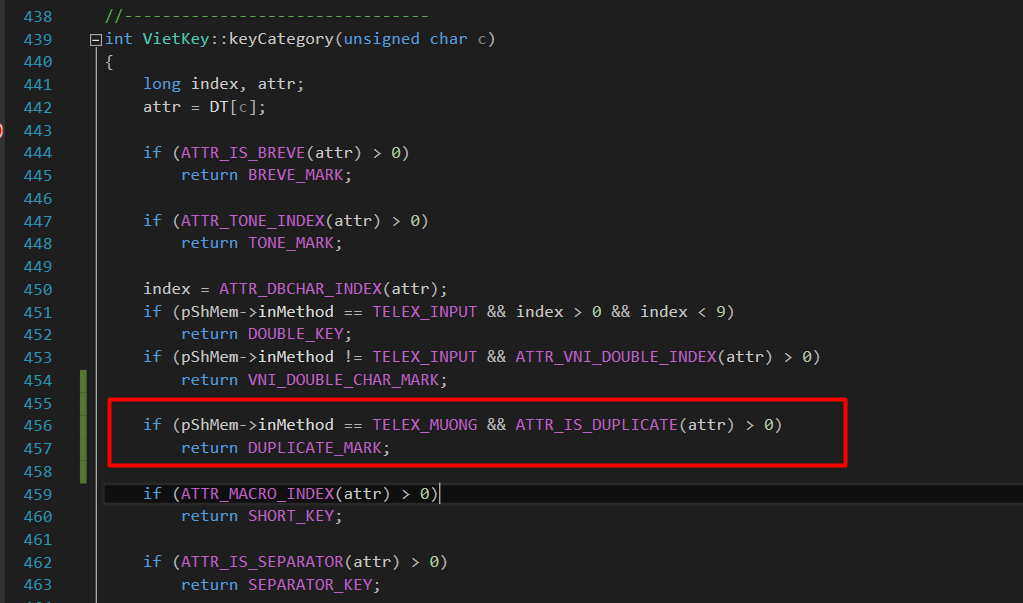
Như các bạn thấy, hiện tại còn thừa Bit số 24 chưa được sử dụng để lưu thông tin gì cả, tôi sẽ sử dụng bit số 24 để lưu giá trị. Nếu bit 24 có giá trị thì tôi sẽ gọi nó là DUPLICATE KEY

Cách định nghĩa như sau. Định nghĩa ID của DUPLICATE KEY

#define DUPLICATE\_MARK 10 Trong vietkey.cpp

Định nghĩa hàm #define ATTR\_IS\_DUPLICATE(x) ((x >> 23) & 0x1) ở trong keycons.h . Hàm này kiểm tra, nếu Bit số 24 mà có giá trị thì nó sẽ là DUPLICATE KEY

Định nghĩa thêm đoạn code sau trong hàm keyCategory()



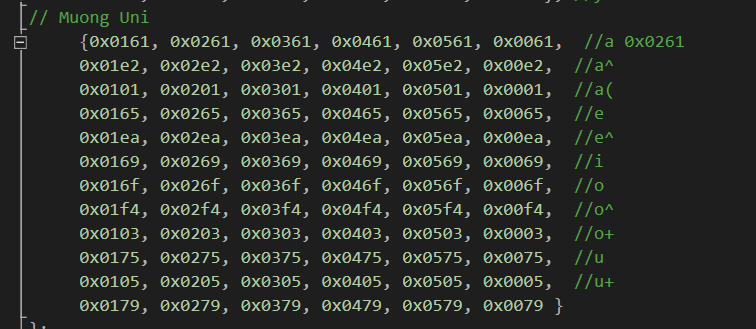
1. **ĐỊNH NGHĨA BẢNG MÃ CỦA TIẾNG MƯỜNG**

Bảng chữ cái tiếng Mường có đặc điểm sau, vì các kí tự có dấu của tiếng Mường không được hỗ trợ sẵn trong bảng mã unikey dựng sẵn. Nên bắt buộc phải dùng Unicode tổ hợp để thể hiện.

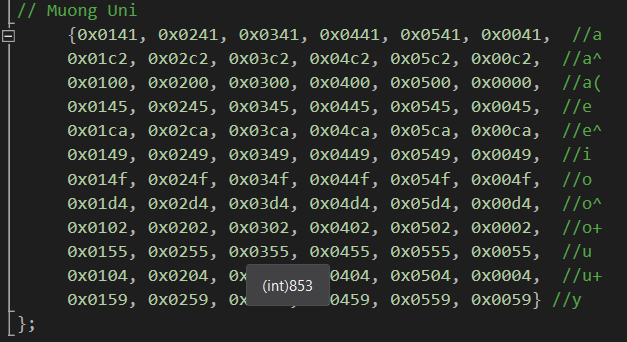
Bảng mã Unicode tổ hợp được tạo bởi các ký tự trong bảng chữ cái tiếng việt cơ bản + các ký tự dấu.

Vì vậy cần định nghĩa 3 bảng

1. Bảng tổ hợp các chữ cái tiếng Mường viết thường UniVnL[][72] ở encode.cpp

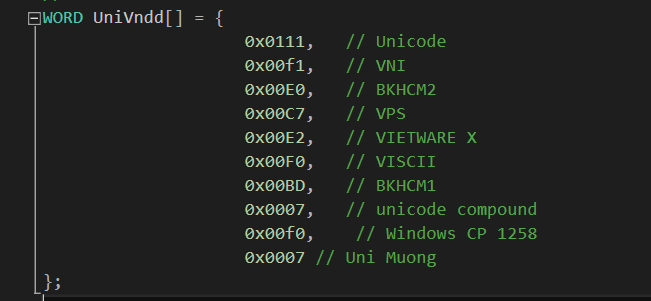
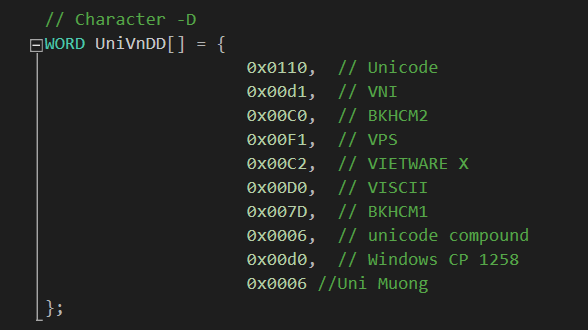


1. Bảng tổ hợp các chữ cái tiếng Mường viết hoa UniVnH[][72] ở encode.cpp



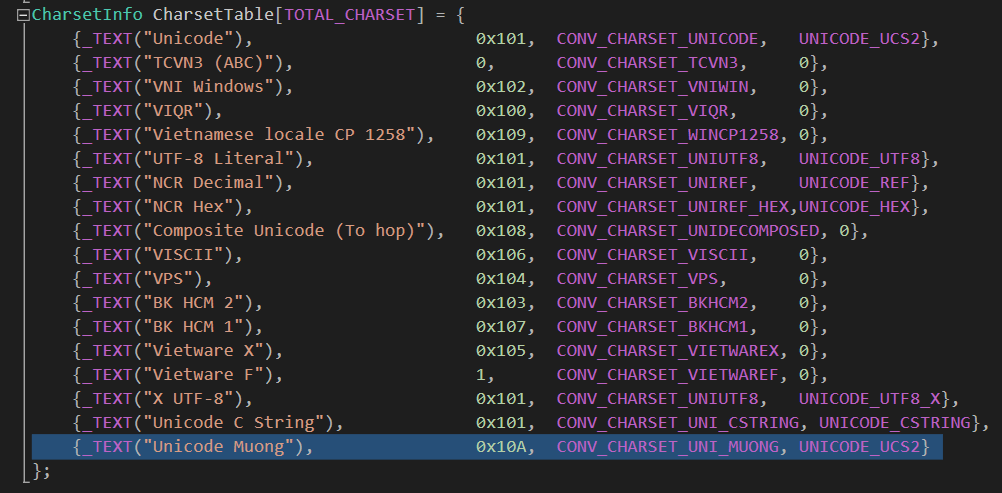
1. Bảng dấu của các kí tự tiếng mường UnicodeMuongTone[]. Được định nghĩa ở vietkey.cpp

  
4. Bảng chữ D

Sau khi định nghĩa 3 bảng mã. Ta cần Thêm ID cho bảng mã vừa tạo.   
CharsetTable[TOTAL\_CHARSET].

1. Tăng giá trị của TOTAL\_CHARSET thêm 1 đơn vị
2. Định nghĩa #define CONV\_CHARSET\_UNI\_MUONG 7 trong vnconv.h
3. Định nghĩa như sau



Các tham số được định nghĩa như sau

<Tên Bảng mã> , <Vị trí của Bảng mã trong UniVnH[] và UniVnL[] + thêm 0x101>, <ID của bảng mã> <Chuẩn mã hóa>

Vì chúng ta sử dụng kí tự 2 byte, nên ta cần tăng CONV\_TOTAL\_DOUBLE\_CHARSETS trong vnconv.h thêm 1 đơn vị.

Giờ cần tạo tiếp 1 class UnicodeMuongCharset trong charset.h & và định nghĩa nó trong charset.h

Cách định nghĩa tương đương UnicodeCompCharset vì cũng là bảng mã unicode tổ hợp.

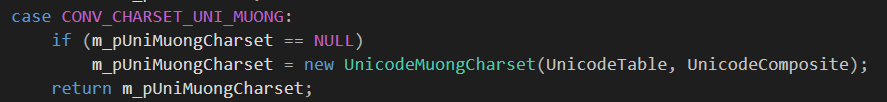
1. Trong CvnCharsetLib của charset.h tạo 1 con trỏ

UnicodeMuongCharset \*m\_pUniMuongCharset;

1. Trong hàm getVnCharset() của CvnCharsetLib thêm swicth case cho trường hợp nếu bảng mã được chọn là bảng mã tiếng Mường

case CONV\_CHARSET\_UNI\_MUONG:

Nếu là tiếng Mường thì gọi hàm khởi tạo của bảng mã Mường



1. **ĐỊNH NGHĨA KIỂU GÕ TELEX MƯỜNG**

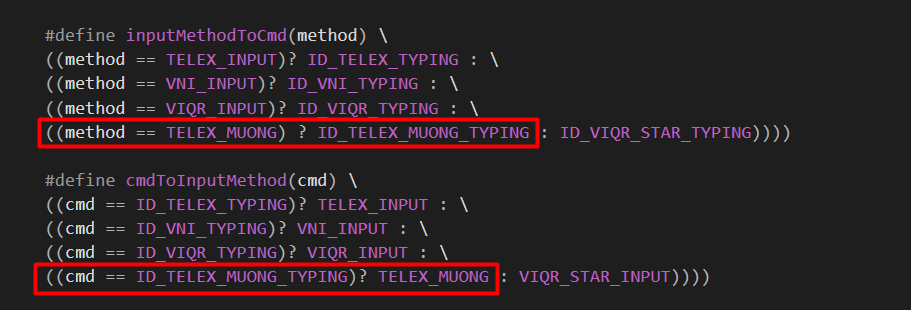
Sau khi có bảng mã Mường, ta cần định nghĩa cách gõ riêng cho tiếng Mường, vì tiếng Mường khác tiếng việt.

1. Các điểm khác biệt cơ bản của Telex Mường với Telex Việt

* Chữ Q nhân đôi nguyên âm trong từ
* Trong tiếng Mường có chữ W sử dụng khá phổ biến (W vừa làm dấu, vừa là phụ âm, vừa là âm đệm nên không được để phím W như hiện tại.
* Bổ sung cách gõ tắt mới
* Thay thế 1 số luật
* Vừa giữ kiểu gõ Telex cũ và Bổ sung thêm Telex Mường để có thể sử dụng song song cả 2 kiểu gõ Telex nếu có nhu cầu

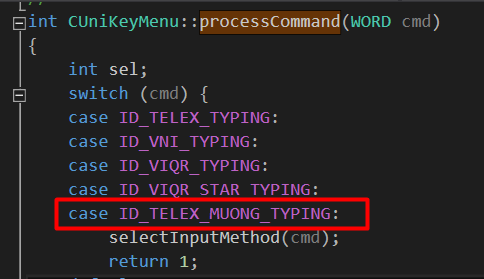
Cách làm lần lượt như sau:

1. Trong keycons.h định nghĩa TELEX\_MUONG có vị trí trong bảng các kiểu gõ bằng 4, và sau đó tăng số INPUT\_METHOD thêm 1 đơn vị
2. Trong resource.h định nghĩa ID cho kiểu gõ TELEX\_MUONG bằng cách định nghĩa ID\_TELEX\_MUONG\_TYPING 40013
3. Trong Ukmenu.cpp viết định nghĩa bổ sung cho 2 hàm

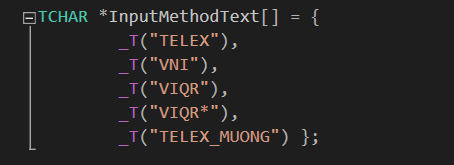


Hàm này sẽ phụ trách việc ánh xạ ID được chọn trong bảng các kiểu gõ với ID của kiểu Gõ

1. Trong processCommand() của ukmenu.cpp , bổ sung thêm trường hợp. Nếu id được chọn là ID của Telex\_Mường



1. Trong keydlg.cpp InputMethodText[] Bạn cần thêm Telex Mường vào mảng input, theo đúng thứ tự khai báo bên trên



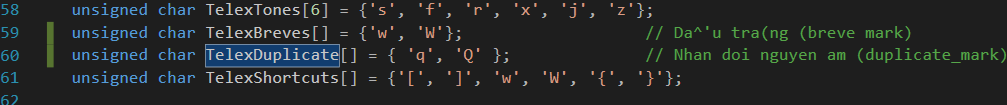
1. BuildTelexWithMuongMethod là 1 hàm trong endcode.cpp Đây là hàm dùng để định nghĩa cách gõ TELEX\_MUONG, về cơ bản thì nó sẽ đảm nhiệm vai trò quyết định, nó sẽ quyết định phím nào đóng vai trò gì, phím nào không.

Từ đầu đến giờ chúng ta đã tạo bảng mã, tạo cách gõ, viết hàm kiểm tra các bit trong 32 bit xem bit nào là vai trò gì. Nhưng chưa viết hàm gán giá trị cho các bit.

Ví dụ: tôi đã kiểm tra bit 24 của 1 phím, nếu phím đó có giá trị thì sẽ gọi nó là Duplicate Key. Nhưng hiện tại ta cần thêm 1 hàm để gán giá trị bit cho phím đó nữa.

Giờ công việc cần làm là định nghĩa xem phím nào sẽ là phím Breve, Phím nào sẽ là phím Duplicate, Phím nào sẽ là phím Double Key…..

Đầu tiên, để quy định xem phím nào sẽ là phím duplicate, tôi định nghĩa 1 mảng TelexDuplicate ở trong endcode.pp để định nghĩa các phím nào sẽ là phím Duplicate



Hoặc các phím tắt nào sẽ là phím tắt trong bộ gõ Mường

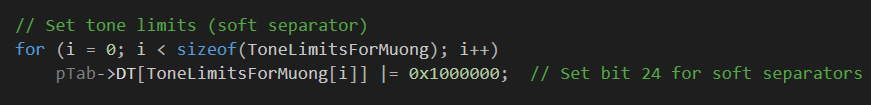
Giới thiệu ý nghĩa của 1 số mảng:

* ToneLimits là mảng giới hạn dấu, nếu trong 1 từ mà từ đó kết thúc bằng những kí tự của ToneLimits thì bạn sẽ không thể thả dấu. Trong tiếng mường, có một số kí tự, ví dụ như w, l có thể đứng ở cuối từ, chính vì vậy tonelimit tuyệt đối không được là các kí tự này.
* Double char : là mảng các phím mà ấn 2 lần sẽ tạo ra các chữ cái , vd: đ, ê, â, ô….
* TelexTones: là mảng các phím mà ấn vào sẽ tạo ra dấu
* WordStop: mảng mà các kí tự này khi gõ ra thì sẽ tương đương việc kết thúc 1 từ

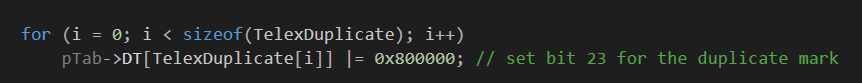
Sau khi định nghĩa xong các mảng ta bắt đầu định nghĩa hàm BuildTelexWithMuongMethod()

* Về bản chất thì Telex Mường cũng giống như Telex thường, chính vì vậy bạn có thể tham khảo cấu trúc hàm của hàm BuildTelexMethod()
* Tuy nhiên phải tùy chỉnh Telex Mường theo đúng những nội dung mảng mà bạn định nghĩa

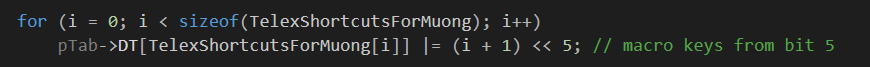
VD:



Hoặc



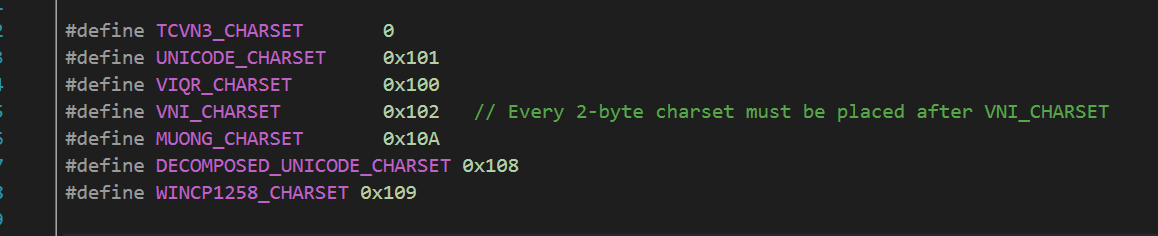
Hoặc



Trong vietkey.cpp

Endcode.cpp

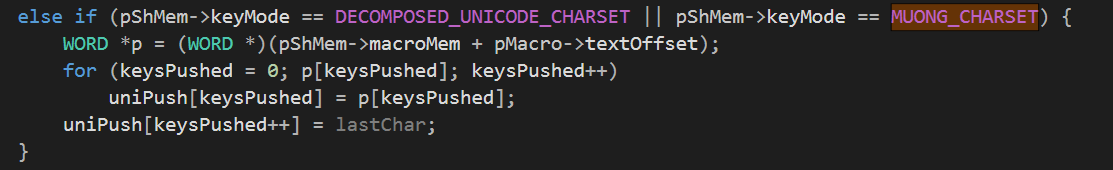
Vào Keycons.h định nghĩa MUONG\_CHARSET . Đây chính là thứ tự của bảng mã của bạn trong encode .cpp



Trong hàm check macro, processBackspace, postProcess, của vietkey.cpp  
PushBuffer của keyhook.cpp

Nếu bạn dùng unicode tổ hợp để xây dựng bảng mã thì hãy xử lý giống DECOMPOSED\_UNICODE\_CHARSET. Còn nếu không, hãy xử lý giống UNICODE\_CHARSET.

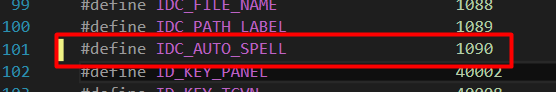
VD :Ở đây vì tôi xử dụng Unicode Tổ hợp để tạo ký tự, nên tôi xử lý giống DECOMPOSED\_UNICODE\_CHARSET . Bằng cách mở tìm kiếm lên, và bất cứ chỗ nào cần xử lý DECOMPOSED\_UNICODE\_CHARSET thì tôi đều bổ sung thêm 1 trường hợp xử lý cho Tiếng Mường như dưới đây



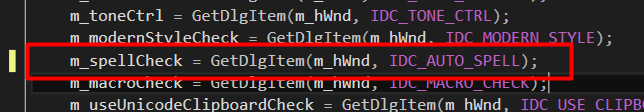
1. **THÊM TÍNH NĂNG CHỈNH SỬA SAU KHI GÕ PHÍM**

Định nghĩa IDC\_AUTO\_SPELL trong resources.h

Đây là 1 cái ID của checkbox trong menu

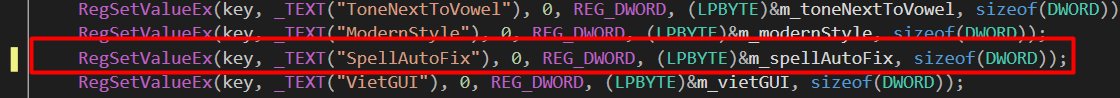


Tạo m\_spellCheck trong keydlg.h và định nghĩa trong onInitDialog, biến này sẽ dùng để kiểm tra xem checkbox đã được check hay chưa

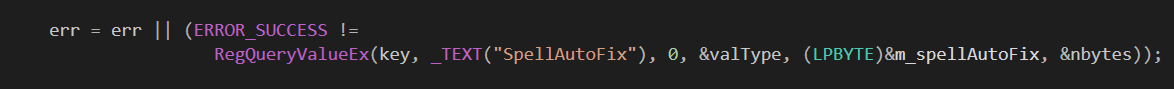


Tạo m\_spellAutoFix trong userpref.h

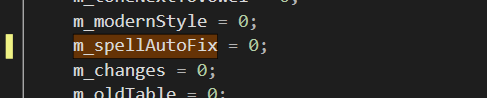
m\_spellAutoFix là 1 biến dùng để lưu giá trị, khi m\_spellCheck thay đổi giá trị thì m\_spellAutoFix cũng sẽ được thay đổi giá trị

Setvalue trong userpref.cpp 

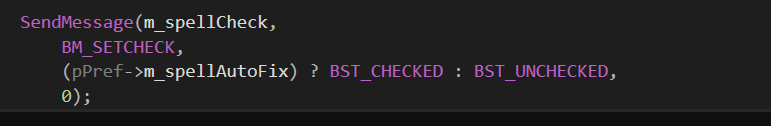
Getvalue trong userpref.cpp



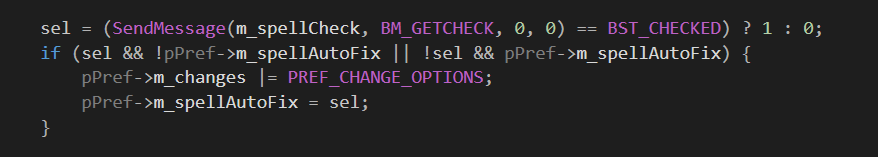
setDefault trong userpref.cpp



m\_spellCheck trong loadDlgInfo



storeDlgInfo



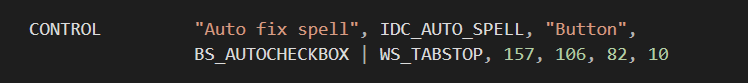
SetHookOptions

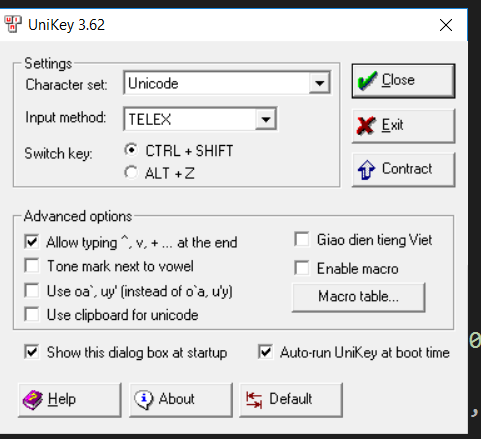
Cuối cùng giá trị của m\_spellAutoFix sẽ được gán cho 1 biến môi trường(). Sau này ta chỉ cần check biến đó có giá trị hay chưa để xác định checkbox đã được đánh dấu hay chưa



Them giao dien trong newkey.rc X2

Ở đây bạn cần vẽ 2 giao diện checkbox cho menu, 1giao diện tiếng việt và 1 giao diện tiếng anh





1. **THÊM PHÍM TẮT CHO BỘ GÕ MƯỜNG**

Vì đặc thù Tiếng Mường sử dụng W như là 1 phụ âm, âm đệm, nên W được sử dụng rất nhiều.

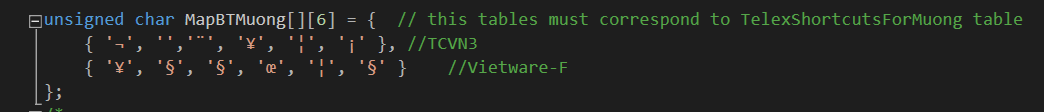
Ví dụ: *kwêl, kwel, khwách. Hay kew, mew, kwew, khwew*

Do đó việc gõ w hiện luôn chữ ư, hoặc gõ uw, aw, ow để viết ư, ă, ơ là không còn hợp lý. Vậy nên ta cần phải tạo phím tắt đặc thù riêng cho tiếng Mường

1. Định nghĩa bảng các phím gõ tắt cho chữ Mường (trong endcode.cpp)



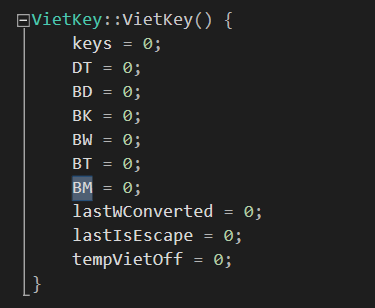
1. Định nghĩa các kí tự gõ tắt tương ứng với cho chữ Mường (trong endcode.cpp)



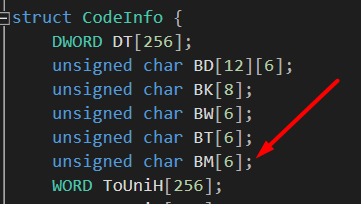
1. Tạo 1 con trỏ thể hiện cho bảng mã này (trong vietkey.h)

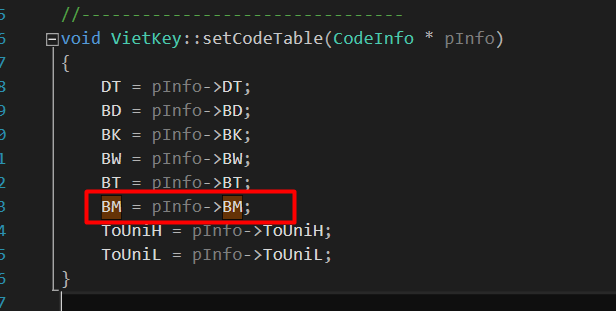


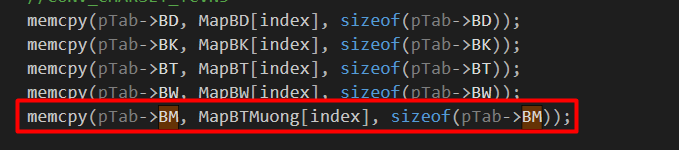
1. Gán BM Null khi được khởi tạo

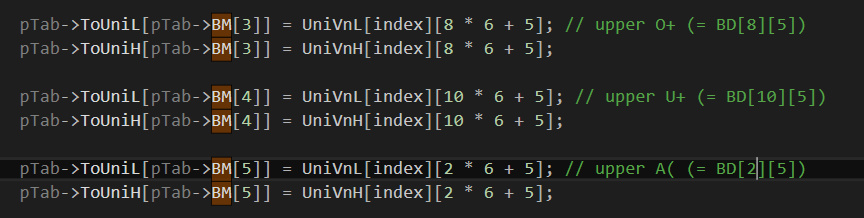


1. Tạo 1 mảng BM trong struct CodeInfo của keyhook.h để lưu các giá trị phím tắt



1. Cho con trỏ BM của vietkey,trỏ vào mảng BM của codeInfo
2. Trong BuildCodeTable. Copy những ký tự thay viết tắt của tiếng Mường sang mảng BM



Cũng trong hàm đó, định nghĩa bổ sung 

Điều này tương đương với việc , nếu phím được bấm mà tạo thành các kí tự lạ kia, thì nó sẽ gán luôn phím đó = kí tự chuẩn Unicode trong bảng mã UniVNL, và UniVnH

1. Tạo Hàm shortKeyForMuong để xử lý trong hàm Process() nếu đó là phím tắt của mường. Có thể bắt chước hàm shortkey của Việt.
2. Trong hàm BuildTelexWithMuongMethod phải gán các phím đó là phím tắt Mường

