

TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO BÀI TẬP LỚN

MÔN HỆ ĐIỀU HÀNH

ĐỀ TÀI: XÂY DỰNG CHƯƠNG TRÌNH ỨNG DỤNG NGÔN NGỮ HỖ TRỢ HỆ ĐIỀU HÀNH WINDOWS

Giảng viên hướng dẫn : TS. Phạm Thanh Bình

Nhóm sinh viên thực hiện :

1. Vũ Thị Hương - 57TH2
2. Lê Xuân Chinh - 57TH2
3. Trần Bá Cương - 57TH2
4. Trương Đức Khang - 57 TH2

Hà Nội, 5/2018

MỤC LỤC

MỞ ĐẦU	3
<u>1.</u> Giới thiệu:.....	3
<u>2.</u> Công cụ sử dụng:	3
CHƯƠNG 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH VÀ HÀM API TRÊN HỆ ĐIỀU HÀNH WINDOWS	4
1.1 Tổng quan về hệ điều hành windows.	4
<u>1.2</u> Tổng quan về tiến trình.	4
1.2.1 Tiến trình là gì?	4
1.2.2 Chương trình ví dụ về tiến trình	4
1.2 Hàm API trên hệ điều hành windows.....	6
1.3.1 Hàm API là gì?	6
1.3.2 Một số hàm API trên hệ điều hành windows	6
CHƯƠNG 2: CHƯƠNG TRÌNH ỨNG DỤNG HỖ TRỢ HỆ ĐIỀU HÀNH VÀ THỰC NGHIỆM	8
2.1 Giới thiệu bài toán chương trình ứng dụng.....	8
2.2 Giải thuật của bài toán.	8
2.3 Mã nguồn chương trình ứng dụng và thực nghiệm.....	9
2.3.1 Mã nguồn chương trình	9
2.3.2 Hình ảnh màn hình thực nghiệm chương trình.....	16
CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	18
3.1 Kết luận.....	18
3.2 Hướng phát triển.....	18
TÀI LIỆU THAM KHẢO	19

MỞ ĐẦU

1. Giới thiệu:

Hệ điều hành là một hệ chương trình hoạt động giữa người dùng và phần cứng của máy tính. Từ đó, cung cấp một môi trường để người sử dụng có thể thi hành các chương trình làm cho máy tính dễ sử dụng hơn, thuận tiện và hiệu quả. Mỗi phiên bản của hệ điều hành đều hỗ trợ mặc định bộ gõ dạng english. Nên để có thể giúp người dùng sử dụng được keyboard của hệ điều hành với unikey tiếng Việt, nhóm đã quyết định lựa chọn đề tài “**Xây dựng ứng dụng unikey hỗ trợ hệ điều hành windown**”.

2. Công cụ sử dụng:

Ngôn ngữ lập trình: C#.

Phần mềm code: Visual studio 2015.

Nội dung bố cục của báo cáo được trình bày trong 3 chương:

Chương 1: Tổng quan về hệ điều hành và hàm API trên hệ điều hành windown.

Chương này trình bày các khái niệm tổng quan logic về hệ điều hành windows. Từ đó, giới thiệu về tiến trình trong hệ điều hành widnows, một số hàm API và công dụng của hàm API trên windows.

Chương 2: Chương trình ứng dụng ngôn ngữ hỗ trợ hệ điều hành và thực nghiệm.

Chương này, trình bày bài toán, giải thuật ứng dụng unikey. Từ đó đưa ra đầy đủ mã nguồn và chú thích cho từng module chức năng và hình ảnh thực nghiệm của ứng dụng.

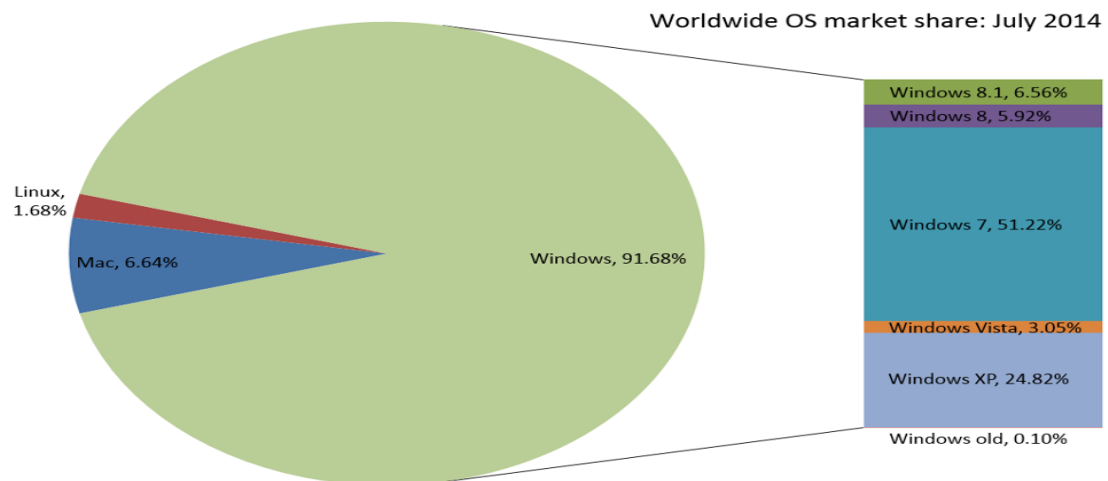
Chương 3: Kết luận và hướng phát triển.

Chương này đưa ra kết luận về những gì đã làm được và rút ra bài học định hướng phát triển ứng dụng cho ứng dụng hỗ trợ hệ điều hành trong tương lai.

CHƯƠNG 1: TỔNG QUAN VỀ HỆ ĐIỀU HÀNH VÀ HÀM API TRÊN HỆ ĐIỀU HÀNH WINDOWS

1.1 Tổng quan về hệ điều hành windows.

Định nghĩa: Hệ điều hành là một chương trình hay một hệ chương trình hoạt động giao tiếp giữa người dùng và phần cứng máy tính. Cung cấp một môi trường để người sử dụng có thể thi hành chương trình, giúp ích cho người dùng trong công việc và cuộc sống. Hệ điều hành bao gồm các chương trình dịch, hệ thống cơ sở dữ liệu,... sử dụng tài nguyên của máy tính để giải quyết các yêu cầu người dùng.



Hình 1: Hình ảnh minh họa sự so sánh giữa các loại hệ điều hành hiện nay

Hệ điều hành windows: Một hệ điều hành đa nhiệm có thể xử lý nhiều chương trình cùng một lúc. Hệ điều hành được phân phối bởi Microsoft.

1.2 Tổng quan về tiến trình.

1.2.1 Tiến trình là gì?

Tiến trình là một thực thể đang thực hiện điều khiển một đoạn mã lệnh riêng không gian, địa chỉ, ngăn xếp và sở hữu một trạng thái giúp thông báo nó đang làm gì (đang chạy, đang chờ, đã đóng, ...).

Tiến trình có 4 thành phần quan trọng: CPU, bộ nhớ, File, Thiết bị nhập xuất.

1.2.2 Chương trình ví dụ về tiến trình

- Mã nguồn code:

+ Khai báo hàm hiện thị danh sách các tiến trình

```
private void loadProcessList()
```

```

    {
        processList = Process.GetProcesses();
+ Nhận tiến trình.
        if(Convert.ToInt32(lblCount.Text) != processList.Length)
        {
            lblList.Items.Clear();
            for(int i = 0; i < processList.Length; i++)
            {
                lblList.Items.Add(processList[i].ProcessName); //Hiện thị danh sách tiến
trình dạng list.
            }
            lblCount.Text = processList.Length.ToString();
        }
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        loadProcessList(); //Load hiện thị danh sách tiến trình.
    }

```

+ Xử lý sự kiện mở một tiến trình.

```

private void btnStart_Click(object sender, EventArgs e)
{
    string text = txtText.Text;
    Process process = new Process();
    process.StartInfo.FileName = text; //Nhập chuỗi nhập vào
    process.Start();
}

```

+ Xử lý sự kiện kill một tiến trình.

```

private void btnStop_Click(object sender, EventArgs e)
{
    processList[lblList.SelectedIndex].Kill();
}

```

```

private void timer1_Tick(object sender, EventArgs e)
{
    loadProcessList();
}

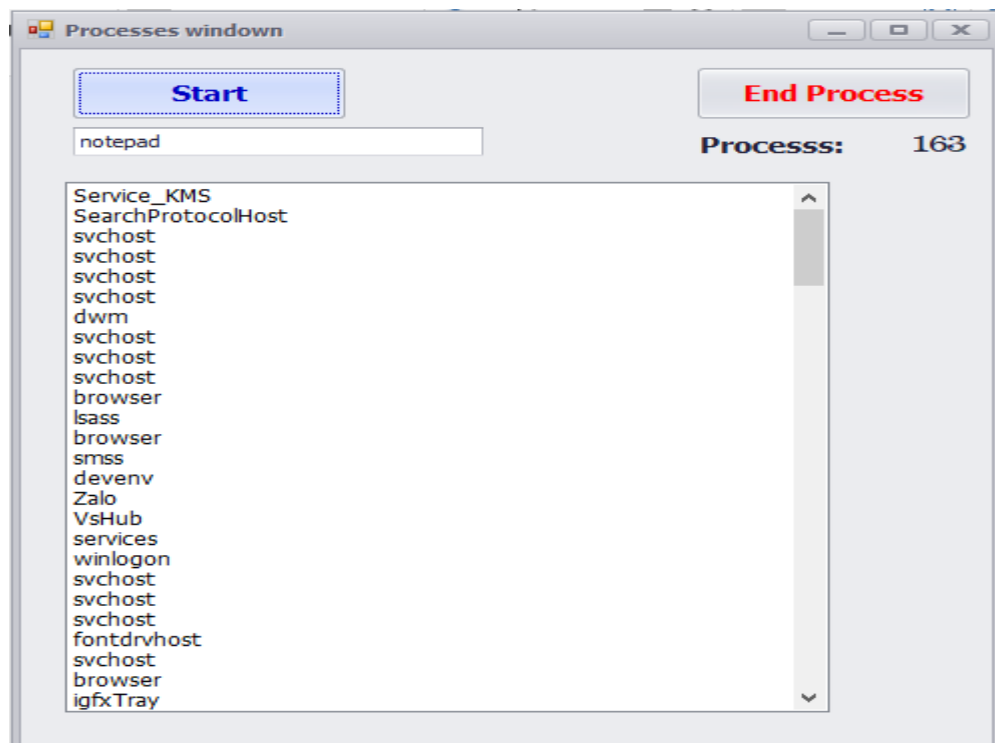
```

```

private void killProcessToolStripMenuItem_Click(object sender, EventArgs e)
{
    processList[lblList.SelectedIndex].Kill();
}

```

- Hình ảnh chạy chương trình:



1.3 Hàm API trên hệ điều hành windows.

1.3.1 Hàm API là gì?

API là viết tắt của Application Programming Interface (giao diện lập trình ứng dụng) phương thức kết nối với các thư viện và ứng dụng khác. Nó chính là một phần mềm giao tiếp được sử dụng bởi các ứng dụng khác nhau giữa chương trình và hệ điều hành.

1.3.2 Một số hàm API trên hệ điều hành windows

- Các API được tổ chức trong 4 DLL của windows:

a. KERNEL32:

Là DLL chính, đảm nhiệm quản lý bộ nhớ, thực hiện chức năng đa nhiệm và những hàm ảnh hưởng trực tiếp đến hoạt động của Windows.

b. USER32:

Thư viện quản lý Windows. Thư viện này chứa các hàm xử lý menu, định giờ, truyền tin, tập tin và nhiều phần không được hiển thị khác của Windows.

c. GDI32:

Giao diện thiết bị đồ họa (Graphics Device Interface). Thư viện này cung cấp các hàm vẽ trên màn hình, cũng như kiểm tra phần biểu mẫu nào cần vẽ lại.

d. WINMM:

Cung cấp các hàm multimedia để xử lý âm thanh, nhạc, video thời gian thực, lấy mẫu, v.v... Nó là DLL 32 bit. (Thư viện 16 bit tên là *MMSYSTEM*)

Ta có thể tìm các tập tin này trong thư mục `\Windows\system`. Ngoài ra, còn có các DLL nhỏ hơn, cũng được dùng phổ biến để cung cấp các dịch vụ đặc biệt cho ứng dụng.

Trên đây là các tên DLL 32 bit. Phiên bản VB4 là bản cuối cùng còn hỗ trợ 16 bit.

- Một số hàm API:

- Declare Function AnyPopup Lib "user32" Alias "AnyPopup" () As Long

Công dụng: Đưa ra chỉ số cửa sổ popup hiện đang tồn tại trên màn hình.

Trị trả về: Integer ~ True (Khác zero) nếu có cửa sổ popup.

- Declare Function AdjustWindowRectEx Lib "user32" Alias "AdjustWindowRectEx" (lpRect As RECT, ByVal dsStyle As Long, ByVal bMenu As Long, ByVal dwExStyle As Long) As Long

Công dụng: Điều chỉnh cửa sổ khi có vùng làm việc client (Không tính kích thước của thanh tiêu đề, đường viền và các phần thêm) được khai báo, khi biết kiểu cửa sổ.

Tham số kèm: LpRect Hình chữ nhật chứa vùng làm việc client. DwStyle Kiểu cửa sổ. BMenu Đưa giá trị True (Khác zero) nếu cửa sổ có trình đơn DwExStyle kiểu cửa sổ mở rộng.

CHƯƠNG 2: CHƯƠNG TRÌNH ỨNG DỤNG HỖ TRỢ HỆ ĐIỀU HÀNH VÀ THỰC NGHIỆM

2.1 Giới thiệu bài toán chương trình ứng dụng.

* Chương trình gõ Tiếng Việt trên Hệ điều hành windows cho phép người dùng máy tính sử dụng bàn phím đưa vào các sự kiện keyboard thành các ký tự Tiếng việt hiện thị trên các thiết bị và chương trình ghi.

* Nguyên tắc chung của bộ gõ là sử dụng Hock bàn phím chặn các thông điệp về bàn phím như: trạng thái, bàn phím, các thông điệp.

* Dùng các giải thuật riêng của mình để xử lý chuỗi đệm đã thu được thành chuỗi Tiếng Việt và xuất ra các thiết bị ghi.

Từ đó, chương trình sẽ sử dụng hỗ trợ hàm API trên hệ điều hành windows tiến trình của chương trình sẽ chạy ngầm định cho phép người dùng gõ unicode Tiếng việt trên các trình soạn thảo ghi.

2.2 Giải thuật của bài toán.

Đầu vào: Sự kiện nhấn người dùng nhấn keyboard. Hệ điều hành sẽ lưu nhật ký của keyboard.

Đầu ra: Cho phép người dùng thực hiện chức năng Unicode trên thiết bị ghi.

Các bước tiến hành:

Bước 1: Bắt sự kiện người dùng nhấn keyboard. Khai báo khởi tạo trạng thái của chương trình.

Bước 2: Thêm các ký tự từ bàn phím vào để tìm từ cuối cùng trong chuỗi để xử lý.

Bước 3: Khai báo sử dụng chức năng Hock (là cơ một cơ chế mà một ứng dụng có thể chặn các sự kiện như bàn phím, chuột,..) và các giá trị hỗ trợ hock.

Bước 4: Xử lý hàm lấy trạng thái từ sự kiện nhấn bàn phím của user. Sử dụng hỗ trợ của hàm API.

Bước 5: Xử lý hàm trả về thông tin xử lý của hệ điều hành.

Bước 6: Xử hàm gọi tới tiến trình console của hệ điều hành.

Bước 7: Xử lý hàm xử lý thao tác dấu câu (kiểu Telex) và lấy ra chuỗi nguyên âm, để xác định vị trí đặt dấu.

Bước 8: Xử lý hàm xóa trong cửa sổ Focus căn cứ vào chuỗi bộ đệm, sau đó gọi tới hàm thủ tục lập bàn phím back để xóa các ký tự ban đầu trong cửa sổ Focus.

Bước 9: Xử lý hiển thị màn hình console và hướng dẫn người dùng sử dụng chương trình ứng dụng.

2.3 Mã nguồn chương trình ứng dụng và thực nghiệm.

2.3.1 Mã nguồn chương trình

+ Khai báo thư viện cho phép gọi tới hàm API của windows.

using Microsoft.Win32;

+ Khai báo các biến để thao tác với nhật ký lịch sử, trạng thái ký tự của keyboard.

```
private static List<int> a = new List<int>() { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
```

```
private static bool startVn = true; //lưu trạng thái mở đóng của gõ tiếng việt.
```

+ Khai báo Hook để xử lý bất sự kiện xử lý của máy tính.

- Hook: là một cơ chế mà một ứng dụng có thể chặn, lấy các sự kiện, như các thông điệp, thao tác chuột, bàn phím.

```
private const int WH_KEYBOARD_LL = 13;
```

+ WH_KEYBOARD_LL: hook cho phép theo dõi sự kiện bàn phím => ứng dụng hook low level keyboard nên cần phải có hằng số xác định kiểu hook.

```
private const int WM_KEYDOWN = 0x0100;
```

+ WH_KEYDOWN: hook cho phép tham số bắt thông điệp keydown khi nhấn phím.

```
private static LowLevelKeyboardProc _proc = HookCallback;
```

+ Gọi hàm này mọi thời gian mỗi khi có một sự kiện keyboard mới được thực thi đưa vào luồng.

```
private static IntPtr _hookID = IntPtr.Zero;
```

+ Khai báo gọi sử dụng tới API "user32.dll" trong window để trả về thông điệp trạng thái.

```
[DllImport("user32.dll", CharSet = CharSet.Auto, ExactSpelling = true, CallingConvention = CallingConvention.Winapi)]
```

+ Khai báo hàm lấy trạng thái của keycode.

```
public static extern short GetKeyState(int keyCode);
```

```
[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
```

```
private static extern IntPtr SetWindowsHookEx(int idHook, LowLevelKeyboardProc lpfn, IntPtr hMod, uint dwThreadId);
```

+ Handle của hook nếu thành công, ngược lại trả về NULL, cài đặt một hook procedure vào một hook chain.

+ Định nghĩa các giá trị biến:

- idHook: loại hook(Xem bảng giá trị trong word):

- lpfn: một con trỏ đến hook procedure. Nếu là global hook thì tham số này phải trỏ đến một hook procedure trong một DLL. Ngược lại thì có thể trỏ đến một đoạn mã đóng vai trò hook procedure trong process hiện tại.

- hMod: handle của DLL chứa hook procedure. Trong trường hợp local hook thì tham số này được đặt là NULL.

- dwThreadId: định danh của thread mà hook procedure sẽ gắn vào. Nếu giá trị này là 0 thì mọi thread sẽ bị ảnh hưởng (global), ngược lại thì chỉ có thread được xác định là bị ảnh hưởng (local).

+ Khai báo gọi tới hàm API.

```
[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
```

```
[return: MarshalAs(UnmanagedType.Bool)]
```

```
private static extern bool UnhookWindowsHookEx(IntPtr hhk); //Giá trị trả về: Một số khác 0 nếu thành công, ngược lại là 0.
```

+ Giải thích tham số:

- hhk: handle của hook sẽ được loại ra khỏi hook chain, lấy từ SetWindowsHookEx.

+ Khai báo gọi tới hàm API.

```
[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
```

```
private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam, IntPtr lParam);
```

+ Chuyển quyền điều khiển cùng các thông tin hook cho hook procedure kế tiếp trong hook chain. Bạn có thể không sử dụng hàm này tuy nhiên chỉ khi nào bạn muốn chặn các hook procedure còn lại trong hook chain.

+ Giải thích tham số:

- hhk: handle của hook lấy được từ SetWindowsHookEx, tuy nhiên tham số này thực sự không cần thiết.

– nCode: tham số này được gọi tên là hook code. Giá trị của nó sẽ quyết định hook procedure có thực hiện các công việc xử lý không. Nếu giá trị này là âm, bạn cần gọi hàm CallNextHookEx để chuyển tới hook procedure kế tiếp và bỏ qua bước xử lý (xem phần hook procedure). Giá trị này có được do hệ thống quyết định và bạn không nên thay đổi giá trị.

– wParam / lParam: các tham số lưu trữ thông tin cần thiết cho hook procedure kế tiếp.

+ Khai báo gọi sử dụng tới hàm API "kernel32.dll" trên windown hàm trả về thông tin về hệ thống hiện tại.

```
[DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]
```

```
private static extern IntPtr GetModuleHandle(string lpModuleName); // lấy ra module đang xử lý
```

+ Xử lý hàm xử lý tiến trình chuyển đổi ký tự thành mã keycode cho máy tính hiệu.

```
private delegate IntPtr LowLevelKeyboardProc(int nCode, IntPtr wParam, IntPtr lParam);
```

```
private static IntPtr SetHook(LowLevelKeyboardProc proc)
```

```
{
```

```
    using (Process curProcess = Process.GetCurrentProcess())
```

```
{
```

```
    using (ProcessModule curModule = curProcess.MainModule)
```

```
{
```

```
        return SetWindowsHookEx(WH_KEYBOARD_LL, proc, GetModuleHandle(curModule.ModuleName), 0);
```

```
    }
```

```
}
```

```
}
```

```
private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)
```

```
{
```

```
    IntPtr a = new IntPtr();
```

```
    if (nCode >= 0 && wParam == (IntPtr)WM_KEYDOWN)
```

```
{
```

```
        int vkCode = Marshal.ReadInt32(lParam);
```

```

        CheckHotKey(vkCode);
        if (startVn == true)
        {
            if (CheckKey(vkCode) == true)
                return lParam;
            else
                return a;
        }
    }
    return a;
}

static bool CheckKey(int vkCode)
{
    capsLock = (((ushort)GetKeyState(0x14)) & 0xffff) != 0; //0x14
    a.Add(vkCode);
    while (a.Count() > 10)
    {
        a.RemoveAt(0);
    }
    if (vkCode == 83 & a[a.Count() - 2] == 65) //á
    {
        SendKeys.Send("{BACKSPACE}");
        if (capsLock != true)
            SendKeys.Send("á");
        else
            SendKeys.Send("Á");
        return true;
    }
}

```

+ Các vkCode khác trong unikey tương tự như đối với từ “á”.

```
}
```

+ Khai báo hàm hookKeyboard run chức năng chương trình.

```
static void HookKeyboard()
```

```
{
```

```
    _hookID = SetHook(_proc);
```

```
    Application.Run();
```

```
    UnhookWindowsHookEx(_hookID);
```

```
}
```

```
static bool isHotKey_openwindow = false;
```

```
static bool isShowing = false;
```

```
static Keys previousKey_openwindow = Keys.Separator;
```

```
static bool isHotKey_Telex = false;
```

```
static bool isTelex = false;
```

```
static Keys previousKey_Telex = Keys.Separator;
```

+ Hàm kiểm tra các key sử dụng chương trình

```
static void CheckHotKey(int vkCode ) {
```

```
if ((previousKey_openwindow == Keys.LControlKey || previousKey_openwindow ==  
Keys.RControlKey) && (Keys)(vkCode) == Keys.K) //Ctrl+K
```

```
    isHotKey_openwindow = true;
```

```
    if (isHotKey_openwindow)
```

```
    {
```

```
        if (!isShowing)
```

```
        {
```

```
            DisplayWindow();
```

```
        }
```

```
    else
```

```
        HideWindow();
```

```

        isShowing = !isShowing;
    }

    previouresKey_openwindow = (Keys)vkCode;

    isHotKey_openwindow = false;

    if (previouresKey_Telex == Keys.LControlKey && (Keys)(vkCode) == Keys.LShiftKey)
    //Ctrl+Shift

        isHotKey_Telex = true;

        if (isHotKey_Telex)
        {
            if (!isTelex)
            {
                startVn = true;

                Menu();
            }
            else
            {
                startVn = false;

                Menu();
            }
            isTelex = !isTelex;
        }

        previouresKey_Telex = (Keys)vkCode;

        isHotKey_Telex = false;

        if ((Keys)(vkCode) == Keys.CapsLock)
        {
            Menu();
        }
    }
}

```

```
[DllImport("kernel32.dll")]
```

```
static extern IntPtr GetConsoleWindow(); //lấy ra màn hình console
```

```
[DllImport("user32.dll")]
```

+ Xử lý hàm ẩn hiện windows

```
static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);
```

+ Giải thích các tham số:

* hWnd - Giá trị con trỏ của màn hình hiện tại.

* nCmdShow - giá trị của hàm để thực hiện vd: 0 là mã ẩn cửa sổ - 5 là mã hiện

+ Mã ẩn cửa sổ

```
const int SW_HIDE = 0;
```

```
const int SW_SHOW = 5;
```

```
static void HideWindow()
```

```
{
```

```
IntPtr console = GetConsoleWindow();
```

```
ShowWindow(console, SW_HIDE);
```

```
}
```

```
static void DisplayWindow()
```

```
{
```

```
IntPtr console = GetConsoleWindow();
```

```
ShowWindow(console, SW_SHOW); // hiện window
```

```
}
```

+ Hàm main xử lý giao diện chính của chương trình ứng dụng:

```
static void Menu()
```

```
{
```

```
Console.Clear();
```

```
Console.OutputEncoding = Encoding.UTF8;
```

```
Console.WriteLine();
```

```

        Console.WriteLine("CHƯƠNG TRÌNH ỨNG DỤNG HỖ TRỢ UNIKEY TRÊN
WINDOWN:");

        Console.WriteLine();

        if (startVn == true)

            Console.WriteLine("  Trạng thái: Gõ unicode đang ON.");

        else

            Console.WriteLine("  Trạng thái: Gõ unicode đang OFF.");

        Console.WriteLine("=====");

        Console.WriteLine("  Hướng dẫn sử dụng ứng dụng: ");

        Console.WriteLine("  - Để ON/OFF gõ unicode nhấn tổ hợp: Ctrl+Shift ");

        Console.WriteLine("  - Để ON/OFF cửa sổ chương trình ứng dụng   : Ctrl+K ");

    }

    static void Main(string[] args)

    {

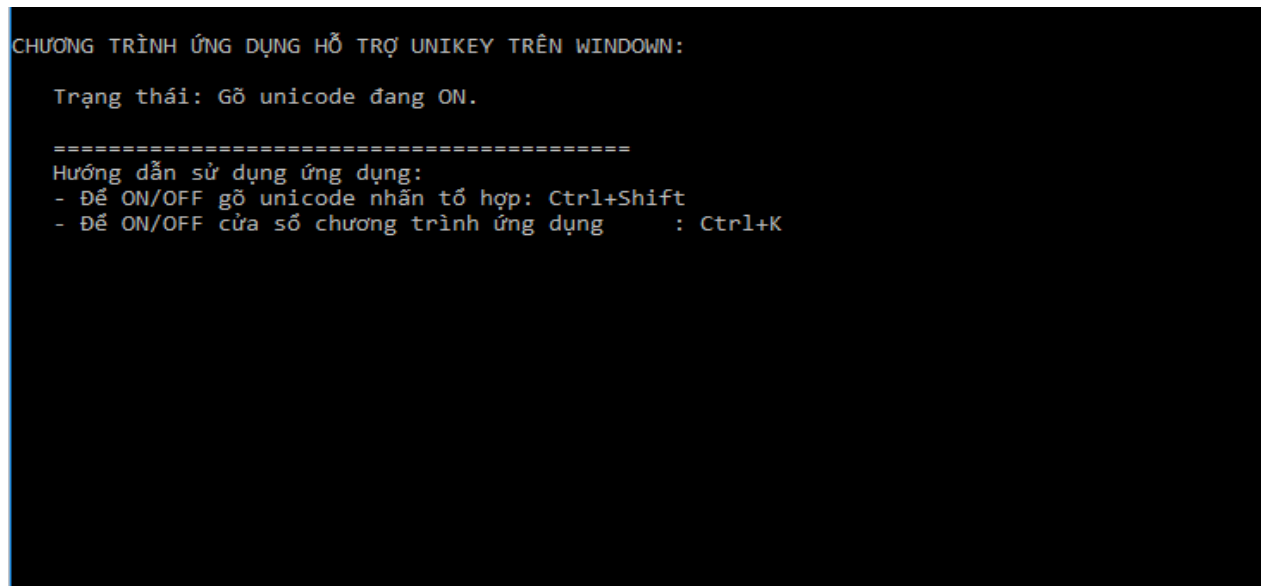
        Menu();

        HookKeyboard();

    }
}

```

2.3.2 Hình ảnh màn hình thực nghiệm chương trình



```

CHƯƠNG TRÌNH ỨNG DỤNG HỖ TRỢ UNIKEY TRÊN WINDOWN:

Trạng thái: Gõ unicode đang ON.

=====
Hướng dẫn sử dụng ứng dụng:
- Để ON/OFF gõ unicode nhấn tổ hợp: Ctrl+Shift
- Để ON/OFF cửa sổ chương trình ứng dụng   : Ctrl+K

```

Hình 3: Hình ảnh On chức năng chương trình ứng dụng

Đầu ra (demo):

Đang học môn hệ điều hành

CHƯƠNG TRÌNH ỨNG DỤNG HỖ TRỢ UNIKEY TRÊN WINDOWN:

Trạng thái: Gõ unicode đang OFF.

=====

Hướng dẫn sử dụng ứng dụng:

- Để ON/OFF gõ unicode nhấn tổ hợp: Ctrl+Shift
- Để ON/OFF cửa sổ chương trình ứng dụng : Ctrl+K

Hình 3: Hình ảnh off chức năng chương trình ứng dụng

Đầu ra (demo):

Ddang hocj moon heej ddieeuf hanhf

CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

3.1 Kết luận.

Bộ gõ tiếng Việt là một loại phần mềm hỗ trợ soạn thảo văn bản bằng tiếng Việt trên máy tính, thường cần phải có phông ký tự chữ Quốc ngữ đã được cài đặt trong máy tính. Bộ gõ tiếng Việt của chúng em sẽ hỗ trợ một bảng mã là bảng mã unicode.

Unicode là bộ mã chuẩn quốc tế được thiết kế để dùng làm bộ mã duy nhất cho tất cả các ngôn ngữ khác nhau trên thế giới, kể cả các ngôn ngữ sử dụng ký tự tượng hình phức tạp như tiếng Trung, tiếng Thái... Vì điểm ưu việt đó, Unicode đã và đang từng bước thay thế các bộ mã truyền thống, kể cả bộ mã tiêu chuẩn ISO 8859 và hiện đang được hỗ trợ trên rất nhiều phần mềm cũng như các trình ứng dụng.

Sau thời gian nghiên cứu học hỏi, nhóm em đã xây dựng thành công chương trình ứng dụng hỗ trợ ngôn ngữ cho hệ điều hành windows. Với chức năng on, off chạy ngầm trên tiến trình của hệ điều hành. Tuy nhiên chương trình không tránh được nhiều sai sót, nhóm em rất mong nhận được góp ý từ thầy. Nhóm em chân thành cảm ơn thầy nhiều.

3.2 Hướng phát triển.

Chúng em sẽ cố gắng hoàn thiện và thêm một số bảng mã nữa như VNI windows, NCR hex, VCSI II... Đồng thời, áp dụng kiến thức môn học để phát triển nhiều ứng dụng hơn nữa để hỗ trợ cho hệ điều hành windows.

TÀI LIỆU THAM KHẢO

- [1] <http://dhthuyloi.blogspot.com/2014/02/tai-lieu-he-ieu-hanh.html>
- [2] <http://www.mediafire.com/file/dz5gr9vk5p9a0yr/Modern+Operating+Systems+2Nd+Ed+By+Tanenbaum+%28Prentice+Hall%29.chm>
- [3] <https://msdn.microsoft.com/en-us/library/>
- [4] Petzold, C. (2002). *Programming Microsoft Windows with C#*(pp. 89-98). Redmond, Washington: Microsoft Press.