**Report on Keylogger Implementation Using**

**Object-Oriented Programming in C++**

Muhammad Saad and Omer Imran Siddiqui

Department of Cyber Security

CS112: Object Oriented Programming

Ahmad Nawaz

May 09, 2025

## 1. Introduction

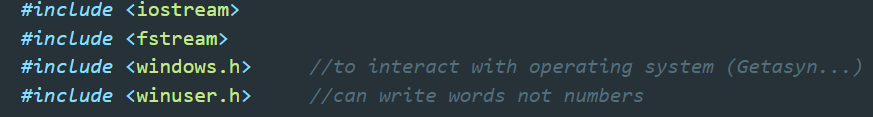
This report presents a simple implementation of a keylogger in C++ using object-oriented programming (OOP) principles. The keylogger captures and logs every keystroke made by the user into a file named testing.txt. Emphasis is laid on OOP concepts like abstraction, encapsulation, inheritance, and polymorphism.

## 2. Objective

To demonstrate how keylogging can be achieved through clean OOP design in C++, highlighting the use of abstract classes, dynamic binding (runtime polymorphism), and encapsulation.

## 3. System Requirements

* Windows Operating System (as it uses windows.h and winuser.h)
* C++ compiler (e.g., MinGW, MSVC)

windows.h and winuser.h are essential for interacting with the OS and accessing the virtual key codes used to detect key presses.

## 5. OOP Concepts Used

### 5.1 Abstraction

* The Logger class provides an abstract interface through the pure virtual function log(), hiding the implementation details from the user.

### 5.2 Inheritance

* FileLogger inherits from Logger, allowing different logging strategies (e.g., logging to console, network, or file) to be easily added by extending Logger.

### 5.3 Encapsulation

* FileLogger encapsulates the file stream object logfile and ensures proper file handling via constructor and destructor.

### 5.4 Polymorphism (Runtime)

* A pointer of type Logger\* is used in KeyLogger, and it calls log() on this pointer. At runtime, the correct implementation (FileLogger::log) is called due to the virtual function.

## 6. Code Walkthrough

### Main Function:

A screen shot of a computer code

AI-generated content may be incorrect.

* Hides the console window using FreeConsole().
* Creates a FileLogger instance.
* Passes it to KeyLogger using a pointer to base class Logger.
* Begins logging using start() method.

### Logger Class (Abstract Base Class):

**A computer code with colorful text

AI-generated content may be incorrect.**

* + Defines a pure virtual function log().
  + Declares a virtual destructor.

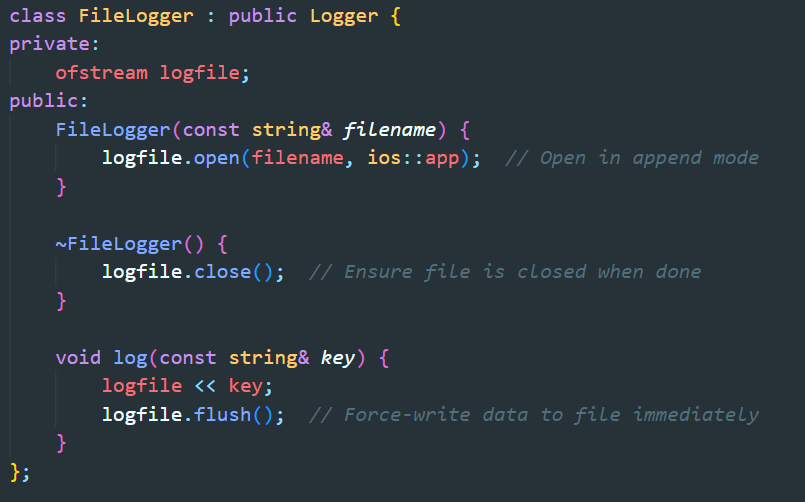
### FileLogger Class:

class FileLogger : public Logger {

private:

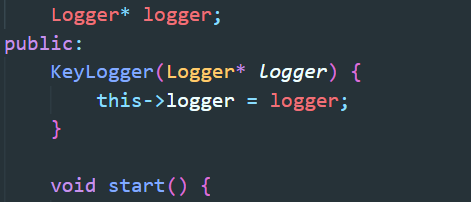
ofstream logfile;

public:

Implements the log() function.

* Opens the file in append mode.
* Flushes the buffer to ensure immediate writing.

### KeyLogger Class:



* Uses Logger\* to allow runtime polymorphism.
* Maps key codes to readable strings.
* Continuously checks key states using GetAsyncKeyState().

## 7. Functional Flow

1. main() runs and hides the terminal.
2. A FileLogger object is created and passed to KeyLogger.
3. KeyLogger::start() enters an infinite loop.
4. It checks key states from 0 to 254.
5. When a key press is detected (& 0x1), it maps and logs it.
6. The file stays open until the object is destructed (program exit).

## 8. Output

* All keystrokes are logged in testing.txt.
* Example log:



9. Security & Ethical Consideration:

This code is meant for educational use only. Deploying or distributing keyloggers without explicit consent is unethical and illegal in many jurisdictions.

10. Conclusion:

This project demonstrates a clean and maintainable way to build a keylogger using core OOP principles in C++. The use of abstract base classes, encapsulation of file I/O, and runtime polymorphism ensures a flexible and extensible design.

**End of Report**