Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Разработка приложения для запуска программы по нажатию сочетания клавиш

Курсовой проект по учебной дисциплине «Системное программирование»

по специальности 09.03.01 Информатика и вычислительная техника  
ИЭИС.КП 2092зу. 004.457ПЗ

|  |  |
| --- | --- |
|  | Руководитель  / И. Ю. Кулаков  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |
|  | Студент группы 2092зу  / Н.В. Кузнецов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

# Содержание

Введение 4

1 Постановка задачи 6

1.1 Обоснование необходимости разработки 6

1.2 Технико-математическое описание задачи 6

1.3 Анализ предметной области 6

1.4 Требования к программе 7

1.5 Обоснование проектных решений 7

1.6 Обзор и анализ существующих программных систем 7

1.7 Выводы предпроектного исследования 8

2. Теоретическая часть 9

2.1 История научных исследований по выбранной теме 9

2.2 Определение и анализ ключевых терминов 9

2.3 Актуальные взгляды на выбранную проблему 10

3 Практическая часть 11

3.1 Анализ задачи 11

3.2 Описание логической структуры 11

3.3 Разработка программы для работы 13

3.4 Сравнительная характеристика работы 22

Заключение 23

Список литературы 24

Приложение А 25

Приложение Б 30

Приложение В 31

Приложение Г 33

Приложение Д 67

# Введение

Задачей курсового проекта является разработка приложения для запуска программ по нажатию сочетания клавиш в операционной системе Windows. В процессе работы над проектом должна быть создана программа, с помощью которой пользователь сможет получать быстрый доступ к программам с помощью горячих клавиш. Программы для запуска и сочетания клавиш заранее назначаются пользователем в интерфейсе приложения. Назначением разрабатываемой программы является ускорение доступа к часто используемым приложениям.

Потенциальным пользователем приложения является любой пользователь компьютера, которому требуется получать быстрый доступ к каким-либо программам. К таким пользователям можно отнести офисных сотрудников, IT‑специалистов, геймеров и другие группы людей, которые проводят много времени за компьютером.

В современном мире люди тратят все больше времени на взаимодействие с различными техническими устройствами. Часто для выполнения рутинной задачи пользователю приходится совершать определенную последовательность действий. Доступ к приложениям по комбинации клавиш может сократить время, используемое на простые задачи. Стандартные решения ограничены в функциональности и не позволяют настраивать горячие клавиши для запуска сторонних программ.

Целью данного проекта является разработка приложения для Windows, позволяющего пользователю настраивать и использовать сочетания клавиш для быстрого запуска программ.

Задачи:

- проведение анализа существующих решений и технологий для реализации приложения;

- разработка приложения, включая модули для обработки горячих клавиш и управления запуском программ;

- создание удобного пользовательского интерфейса;

- тестирование разработанного приложения;

- подготовка документации.

# 1 Постановка задачи

# 1.1 Обоснование необходимости разработки

В соответствии с заданием на курсовой проект, требуется разработать приложение, способное запускать сторонние программы по нажатию комбинации клавиш. Программа должна иметь интерфейс для настройки горячих клавиш.

# 1.2 Технико-математическое описание задачи

Для решения задачи требуется перехватывать нажатия клавиш пользователя. Так как интерфейс программы во время ее работы должен быть скрыт, для захвата клавиатуры потребуется использовать глобальный хук. Графический интерфейс можно реализовать на основе Windows Forms, который является частью Microsoft .NET Framework. Наиболее подходящими для разработки такого приложения являются языки C# и C++.

# 1.3 Анализ предметной области

Среди пользователей приложения можно выделить две группы: администраторов и конечных пользователей. Администратор настраивает горячие клавиши и выполняет их сопоставление с исполняемым файлом целевой программы. Пользователь же, используя горячие клавиши, заданные администратором, получает быстрый доступ к нужным приложениям. В большинстве случаев функции пользователя и администратора будет выполнять один человек.

# 1.4 Требования к программе

Для корректной работы программы является ПК под управлением ОС Windows 7 или выше с установленным пакетом Microsoft .NET 6.0.

# 1.5 Обоснование проектных решений

1.5.1 Обоснование выбора языков программирования

Наиболее подходящим языком программирования для разработки приложения является C#, так как он является основным языком программирования для платформы .NET.

1.5.2 Инструментальные средства

Платформа .NET отлично подходит для решения поставленных задач, так как включает в себя библиотеки для работы с горячими клавишами. Редактор Notepad++ будет использоваться для ручной проверки текстовых файлов.

1.5.3 Обоснование выбора среды программирования

Для разработки программы будет использоваться Microsoft Visual Studio, потому что она является официальной средой разработки для C# и .NET, что обеспечивает:

- полную совместимость с языком и платформой;

- доступ ко всем функциям и библиотекам .NET.

1.5.4 Информационное обеспечение

Сторонних программных средств для работы приложения не требуется. Для проверки работы приложения подойдет любой исполняемый файл, запуск которого можно проверить.

# 1.6 Обзор и анализ существующих программных систем

Наиболее известными аналогами приложения являются AutoHotkey и Launchy. Обе программы имеют широкий функционал, при этом требуют от пользователя определенных технических знаний для настройки, что затрудняет их использование слабыми пользователями ПК.

# 1.7 Выводы предпроектного исследования

Целью разработки приложения является создание удобного инструмента для быстрого запуска программ. Приложение должно запускаться вместе с системой и иметь интуитивно понятный интерфейс.

Этапы разработки:

- разработка архитектуры приложения;

- создание основного приложения;

- тестирование;

- создание вспомогательного приложения с графическим интерфейсом;

- тестирование;

- разработка документации.

# 2 Теоретическая часть

# 2.1 История научных исследований по выбранной теме

Клавиши-модификаторы появились еще до компьютеров. Пишущая машинка Remington No.2, созданная в 1878 году имела клавишу Shift для переключения прописных и строчных букв. После, разные разработчики добавляли все новые клавиши-модификаторы, которые добавлялись для расширения возможностей клавиатуры. Клавиш-модификаторов становилось все больше вплоть до семи на клавиатуре Space-cadet (Ctrl, Meta, Hyper, Super, Shift, Front, Top).

В 1974 году Ларри Теслер, сотрудник исследовательского центра Xerox в Пало-Альто, создал три самых популярных горячих клавиши – вырезать, копировать, вставить. А в 1983 году Теслер разрабатывал пользовательский интерфейс для Lisa и решил использовать клавиши Z, X, C и V в сочетании с клавишей Apple для обозначения отмены, вырезания, копирования и вставки.

Привычные сейчас Ctrl+Z, Ctrl+X, Ctrl+C и Ctrl+V появились в Microsoft Word еще в Windows 2.0, а с версии 3.1 и в остальных приложениях.

# 2.2 Определение и анализ ключевых терминов

Сочетания клавиш — это комбинации клавиш, предоставляющие альтернативный способ выполнения действий, которые обычно выполняются с помощью мыши.

Автоматизация ПО — это практика создания ПО для сокращения или устранения вмешательства человека в повторяющиеся, трудоемкие ИТ-задачи.

Интерфейс пользователя — интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы.

.NET — это бесплатная кроссплатформенная платформа разработки с открытым кодом. Она может запускать программы, написанные на разных языках, при этом C# является наиболее популярным.

# 2.3 Актуальные взгляды на выбранную проблему

В 2005 году было проведено исследование, где опросили 251 опытного пользователя Microsoft Word. Выяснилось, что большинство из них редко используют сочетания клавиш, предпочитая им панели инструментов и иконки. Позже было проведено второе исследование, где пользователи выполняли обычные команды с использованием меню, панелей инструментов с иконками и сочетаний клавиш. Как и ожидалось, горячие клавиши оказались наиболее эффективными.

Эти исследования показывают, что даже опытные пользователи используют графические интерфейсы неэффективно. Один из возможных способов улучшить эффективность пользователей — это обучение, которое предлагает план по переходу от использования выпадающих меню и панелей инструментов с иконками к выполнению команд с помощью сочетаний клавиш.

# 3 Практическая часть

# 3.1 Анализ задачи

Программа состоит из двух подпрограмм — основной программы и программы-конфигуратора. Основная программа не имеет графического интерфейса, отлавливает сочетания клавиш и запускает приложения. Программа-конфигуратор позволяет пользователю настроить горячие клавиши и выбрать приложения для запуска.

# 3.2 Описание логической структуры

3.2.1 Алгоритм программы

3.2.1.1 Программа-конфигуратор

При запуске приложение проверяет наличие конфигурационного файла hotkeys.cfg и загружает из него данные в массив строк. После этого на основе загруженных данных отображается интерфейс.

В методе draw\_interface программа читает строки из считанного ранее массива и парсит их с помощью регулярного выражения. Для каждой строки, успешно прошедшей проверку, создается элемент AppHotkey, который отображает на форме горячую клавишу, клавишу-модификатор и путь к программе, которая будет запускаться. Если клавиша, полученная из конфигурационного файла, не соответствует допустимому формату (Keys), то она подсвечивается красным цветом.

Пользователь может настроить горячую клавишу нажав на соответствующее поле и введя новую клавишу. При нажатии на кнопку «…» открывается диалоговое окно для выбора программы.

Если нажать кнопку «Добавить», то создается новый элемент AppHotkey, интерфейс обновляется для отображения новой строки.

При нажатии кнопки «Сохранить» обновляется конфигурационный файл программы. Весь список горячих клавиш проверяется на корректность данных. Не прошедшие проверку строки в файл не добавляются.

После нажатия кнопки «Запустить» проверяется наличие основного приложения в автозагрузке, приложение добавляется в автозагрузку при необходимости. Запускается основное приложение Hotkey\_starter.exe, если программа уже была запущена, то выполняется ее перезапуск.

Кнопка «Отмена» восстанавливает интерфейс до последней сохраненной конфигурации, а кнопка «Выход» завершает работу приложения.

3.2.1.2 Основная программа

При запуске программы с помощью метода get\_cfg загружаются данные из конфигурационного файла, инициализируется глобальный хук клавиатуры для отслеживания горячих клавиш и запускается цикл обработки сообщений Application.Run, чтобы программа продолжала работу в фоновом режиме.

В методе get\_cfg проверяется наличие конфигурационного файла. Из файла считываются все строки, парсятся с помощью регулярного выражения. Для каждой строки создается объект Hotkey, состоящий из горячей клавиши, клавиши модификатора и пути к программе, которая будет запускаться. Объект Hotkey добавляется в общий список.

При нажатии любой клавиши на клавиатуре вызывается метод key\_down, который принимает нажатую клавишу и состояние клавиш-модификаторов, после чего ищет полученную комбинацию в списке горячих клавиш. Если совпадение найдено, то запускается соответствующая программа.

3.2.2 Составные части программы и связи между ними

Программа состоит из двух основных частей: основной программы (Hotkey\_starter) и программы-конфигуратора (Hotkey\_binder). Конфигуратор записывает данные о горячих клавишах в конфигурационный файл hotkeys.cfg, выполняет запуск и перезапуск основной программы. Основная же программа считывает конфигурацию из файла hotkeys.cfg и на основе этих данных продолжает работу.

# 3.3 Разработка программы для работы

3.3.1 Входные и выходные данные

Программа-конфигуратор принимает на вход конфигурационный файл hotkeys.cfg и данные, которые вводит пользователь. Вводимые пользователем данные принимаются приложением в виде строк, клавиша-модификатор вводится с помощью выпадающего списка, но в последствии все равно конвертируется в строку для дальнейшей обработки. Ко входным данным также следует отнести клики мышью по разным элементам интерфейса и нажатия с клавиатуры.

На выходе конфигуратора получается новый конфигурационный файл hotkeys.cfg, информация о настроенных горячих клавишах, отображаемая в элементах формы, файл ярлыка для запуска основной программы и сигнал системе на запуск/остановку основной программы при необходимости.

Входные и выходные данные для программы-конфигуратора отображены ниже (таблица 1).

Таблица 1 — Входные и выходные данные Hotkey\_binder

|  |  |  |  |
| --- | --- | --- | --- |
| Тип данных | Название | Описание | Пример |
| Входные данные | | | |
| Конфигурационный файл | hotkeys.cfg | Файл, содержащий настройки горячих клавиш в формате: Модификатор+Клавиша="Путь к программе" | Ctrl+A="C:\Program Files\Notepad++\notepad++.exe" |
| Действия пользователя | Нажатие кнопок и ввод данных | Пользователь взаимодействует с интерфейсом: добавляет, редактирует или удаляет горячие клавиши | Нажатие кнопки "Добавить", ввод клавиши Ctrl+A, выбор программы через openFileDialog |
| Выходные данные | | | |
| Обновленный конфигурационный файл | hotkeys.cfg | Файл, обновленный после сохранения настроек пользователем | Ctrl+A="C:\Program Files\Notepad++\notepad++.exe" |
| Запуск программы Hotkey\_starter | start\_program | Запуск основной программы | Запуск Hotkey\_starter.exe |
| Ярлык в автозагрузке | Hotkey Starter.lnk | Ярлык в папке автозагрузки | C:\Users\<User>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\Hotkey Starter.lnk |

Входными данными основной программы являются конфигурационный файл hotkeys.cfg и перехваченные с клавиатуры нажатия. На выходе получается сигнал на запуск нужного приложения (таблица 2).

Таблица 2 — Входные и выходные данные Hotkey\_starter

|  |  |  |  |
| --- | --- | --- | --- |
| Тип данных | Название | Описание | Пример |
| Входные данные | | | |
| Конфигурационный файл | hotkeys.cfg | Файл, содержащий настройки горячих клавиш в формате: Модификатор+Клавиша="Путь к программе" | Ctrl+A="C:\Program Files\Notepad++\notepad++.exe" |
| Нажатия клавиш | key\_down | Нажатия клавиш, отслеживаемые глобальным хуком | Нажатие Ctrl+A |
| Выходные данные | | | |
| Запуск программы | open\_file | Запуск программы, указанной в конфигурационном файле по нажатию горячей клавиши | Запуск Notepad++ по нажатию Ctrl+A |

3.3.2 Используемые методы

В конфигураторе используются как методы класса Hotkey\_binder, отраженные в таблице Б.1.

Также конфигуратор использует и методы класса AppHotkey, описание которых можно увидеть в таблице 3.

Таблица 3 — Методы класса AppHotkey

|  |  |
| --- | --- |
| Метод | Описание |
| AppHotkey | Конструктор класса. Принимает порядковый номер, который будет использоваться для позиционирования на форме и ссылку на родительскую форму. Вызывает методы для создания и настройки элементов интерфейса. |
| add\_modifier | Создает выпадающий список ComboBox для выбора модификатора, устанавливает его параметры и добавляет на форму. |
| add\_plus\_sym | Добавляет символ «+» на форму. |
| add\_key | Создает текстовое поле TextBox для ввода клавиши, устанавливает его параметры и помещает на форму. Подключает обработчики событий. |
| add\_path | Создает текстовое поле TextBox для отображения пути к файлу программы. Устанавливает параметры, добавляет на форму. |
| add\_file | Создает кнопку Button для выбора программы через openFileDialog. Устанавливает параметры, добавляет на форму. Подключает обработчик события нажатия кнопки. |
| Delete | Удаляет все элементы интерфейса, связанные с текущим экземпляром класса. |
| check\_key | Проверяет текст в поле key на соответствие допустимым значениям. |

В основной программе используются методы класса Hotkey\_starter. Они описаны в таблице 4.

Таблица 4 — Методы класса Hotkey\_starter

|  |  |
| --- | --- |
| Метод | Описание |
| Main | Точка входа в программу. Создает экземпляр класса Hotkey\_starter, запускает загрузчик конфигурации get\_cfg. Инициализирует глобальный хук GlobalKeyboardHook для отслеживания нажатия клавиш. Запускает цикл обработки сообщений Application.Run для работы программы в фоновом режиме. |
| get\_cfg | Загружает конфигурацию из файла hotkeys.cfg. Парсит полученные строки с помощью регулярного выражения. Для каждой строки создает по объекту Hotkey и добавляет его в список hotkeys. |
| check\_cfg | Проверяет наличие файла hotkeys.cfg и создает его, если файл отсутствует. |
| key\_down | Метод обработки горячих клавиш. Вызывается при нажатии любой клавиши. Определяет состояние клавиш-модификаторов и сопоставляет полученную комбинацию клавиш со всеми членами списка hotkeys. Если совпадение найдено, происходит запуск соответствующей программы. |
| open\_file | Запускает программу по указанному пути используя параметры, заданные с помощью класса ProcessStartInfo. В случае возникновения ошибки запуск программы не происходит. |

Кроме методов класса Hotkey\_starter, программа также использует для работы методы класса GlobalKeyboardHook, с их описанием можно ознакомиться в таблице 5.

Таблица 5 — Методы класса GlobalKeyboardHook

|  |  |
| --- | --- |
| Метод | Описание |
| GlobalKeyboardHook | Конструктор класса. Вызывает метод SetHook. |
| SetHook | Устанавливает глобальный хук клавиатуры. Возвращает идентификатор хука \_hookID, используя функцию SetWindowsHookEx из Windows API. |
| HookCallback | Обратный вызов при нажатии клавиши. Проверяет, является ли событие нажатием клавиши. Определяет нажатую клавишу и состояние модификаторов. Вызывает событие KeyDown, передавая информацию о нажатой клавише и модификаторах. |
| SetWindowsHookEx | Функция Windows API. Устанавливает глобальный хук для отслеживания клавиш. |
| CallNextHookEx | Передает информацию о событии следующему хуку в цепочке. |
| GetModuleHandle | Возвращает дескриптор модуля текущего процесса. |

3.3.3 Тестирование

В ходе разработки приложения проводилось ручное тестирование функционала программы и обрабатывались различные исключения. Например, при отсутствии конфигурационного файла программа создает новый. Если какие-то строки в файле не соответствуют заданному шаблону, то они игнорируются. В случае недопустимого наименования клавиши в hotkeys.cfg пользователь будет уведомлен об этом цветовым выделением проблемной области (рисунок 1). Если пользователь не предпримет действий для исправления ошибки, то некорректная клавиша будет удалена из конфигурации при следующем сохранении. Несуществующие пути к исполняемым файлам так же игнорируются. При указании файла, который не является исполняемым, запуск по горячей клавише не произойдет без нарушения работы программы, обработка исключения не требуется.

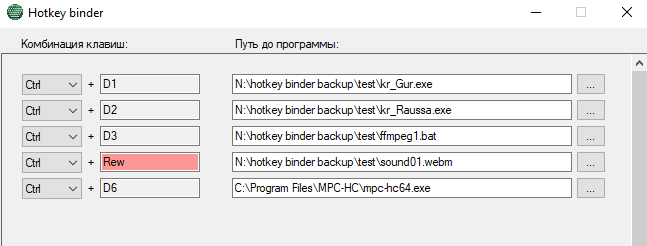


Рисунок 1 — Выделение некорректной области

Для тестирования программы были разработаны тест кейсы (таблицы 6‑13).

Таблица 6 — Тест-кейс, загрузка конфигурации

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU01 | |
| Заголовок | Загрузка конфигурации | |
| Предусловие | Есть программа для проверки работы | |
| Тестовые данные | - | |
| Ожидаемый результат | Конфигурация загружена корректно, горячие клавиши отображаются в интерфейсе | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Создать в папке с исполняемым файлом программы файл hotkeys.cfg с содержимым:  Ctrl+A="C:\Program Files\Notepad++\notepad++.exe"  Alt+S="C:\Windows\System32\calc.exe" | Файл создан |
| 2 | Запустить программу Hotkey Binder | Открылось главное окно программы, в интерфейсе отображаются две строки с горячими клавишами:  Ctrl+A для запуска Notepad++  Alt+S для запуска калькулятора |

Таблица 7 — Тест-кейс, добавление новой горячей клавиши

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU02 | |
| Заголовок | Добавление новой горячей клавиши | |
| Предусловие | Есть программа для проверки работы | |
| Тестовые данные | - | |
| Ожидаемый результат | Новая горячая клавиша Ctrl+B добавлена в конфигурацию и отображается в интерфейсе | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Запустить программу Hotkey Binder | Открылось главное окно программы |
| 2 | Нажать кнопку «Добавить» | В интерфейсе программы добавилась новая строка |
| 3 | В новой строке выбрать модификатор Ctrl. Нажать на поле для ввода клавиши. | В поле для модификатора отображается Ctrl. Поле для ввода клавиши изменило цвет на красный, появилось сообщение «Нажмите клавишу» |
| 4 | Нажать клавишу B | Поле ввода клавиши снова стало светло-серым, в нем отображается введенная клавиша |
| 5 | Нажать кнопку «…» и выбрать любой исполняемый файл | Путь к выбранному файлу отображается в интерфейсе программы |
| 6 | Нажать кнопку «Сохранить» | В нижней части окна появилось сообщение об успешном сохранении конфигурации. Строка с комбинацией Ctrl+B и путем до выбранного фала появилась в конфигурационном файле программы |

Таблица 8 — Тест-кейс, редактирование горячей клавиши

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU03 | |
| Заголовок | Редактирование горячей клавиши | |
| Предусловие | Есть программа для проверки работы, конфигурационный файл с настроенной комбинацией Ctrl+A | |
| Тестовые данные | - | |
| Ожидаемый результат | Горячая клавиша изменена на Shift+F1 для запуска Paint | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Запустить программу Hotkey Binder | Открылось главное окно программы |
| 2 | В строке с горячей клавишей Ctrl+A изменить модификатор на Shift, изменить клавишу на F1, изменить путь к программе на C:\Windows\System32\mspaint.exe | Введенные данные отображаются в интерфейсе |
| 3 | Нажать кнопку «Сохранить» | В нижней части окна появилось сообщение об успешном сохранении конфигурации. В интерфейсе программы и файле конфигурации отображаются новые данные с комбинацией Shift+F1 и путем до исполняемого файла mspaint.exe |

Таблица 9 — Тест-кейс, удаление горячей клавиши

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU04 | |
| Заголовок | Удаление горячей клавиши | |
| Предусловие | Есть программа для проверки работы, конфигурационный файл с настроенной комбинацией Ctrl+A | |
| Тестовые данные | - | |
| Ожидаемый результат | Горячая клавиша Ctrl+A удалена из конфигурации и интерфейса | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Запустить программу Hotkey Binder | Открылось главное окно программы |
| 2 | В строке с горячей клавишей Ctrl+A стереть путь до исполняемого файла программы | Строка с путем пуста |
| 3 | Нажать кнопку «Сохранить» | Комбинация Ctrl+A исчезла из интерфейса и конфигурационного файла |

Таблица 10 — Тест-кейс, проверка некорректных данных

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU05 | |
| Заголовок | Проверка некорректных данных | |
| Предусловие | Есть программа для проверки работы | |
| Тестовые данные | - | |
| Ожидаемый результат | Программа не сохраняет некорректные данные | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Запустить программу Hotkey Binder | Открылось главное окно программы |
| 2 | Выбрать модификатор Shift. Оставить поле для ввода клавиши пустым. Нажать кнопку «…» и выбрать любую программу. | Модификатор выбран, путь до файла указан, поле для клавиши пустое |
| 3 | Нажать кнопку «Сохранить» | Из интерфейса исчезла строка с некорректными данными, в конфигурационном файле не появилось новых записей |

Таблица 11 — Тест-кейс, проверка запуска программы через горячую клавишу

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU06 | |
| Заголовок | Проверка запуска программы через горячую клавишу | |
| Предусловие | Есть программа для проверки работы | |
| Тестовые данные | - | |
| Ожидаемый результат | Программа Paint запускается | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Запустить программу Hotkey Binder | Открылось главное окно программы |
| 2 | Добавить горячую клавишу Ctrl+A для запуска Paint | Горячая клавиша отображается в интерфейсе и файле конфигурации |
| 3 | Нажать кнопку «Запустить» | Отобразилось сообщение об успешном применении конфигурации |
| 4 | Нажать сочетание клавиш Ctrl+A | Запустилась программа Paint |

Таблица 12 — Тест-кейс, проверка добавления в автозагрузку

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU07 | |
| Заголовок | Проверка добавления в автозагрузку | |
| Предусловие | Есть программа для проверки работы | |
| Тестовые данные | - | |
| Ожидаемый результат | В папке автозагрузки создан ярлык Hotkey Starter.lnk | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Запустить программу Hotkey Binder | Открылось главное окно программы |
| 2 | Нажать кнопку «Запустить» | Отобразилось сообщение об успешном применении конфигурации |
| 3 | Проверить папку автозагрузки:  C:\Users\<User>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup | Ярлык Hotkey Starter.lnk находится в папке автозагрузки |

Таблица 13 — Тест-кейс, проверка функции отмены

|  |  |  |
| --- | --- | --- |
| Идентификатор | TU08 | |
| Заголовок | Проверка функции отмены | |
| Предусловие | Есть программа для проверки работы | |
| Тестовые данные | - | |
| Ожидаемый результат | Конфигурация не изменилась, интерфейс вернулся к исходному виду | |
| Шаг | Действие | Предполагаемый результат |
| 1 | Запустить программу Hotkey Binder | Открылось главное окно программы |
| 2 | Нажать кнопку «Добавить» | В интерфейсе программы добавилась новая строка |
| 3 | Заполнить произвольными значениями новую строку | Введенные данные отображаются в интерфейсе |
| 4 | Нажать кнопку «Отмена» | Интерфейс вернулся к исходному виду |

3.3.4 Эксплуатационные документы — руководство пользователя

3.3.4.1 Назначение программы

Программа предназначена для получения быстрого доступа к часто используемым приложения с помощью нажатия горячих клавиш. Может использоваться на домашних компьютерах и офисах, как IT-специалистами, так и менее опытными пользователями ПК.

3.3.4.2 Условия выполнения

Для корректной работы программы требуется компьютер под управлением операционной системы Windows версии 7 или выше. На компьютере должен быть установлен пакет Microsoft .NET 6.0.

Для установки достаточно распаковать архив с программой в удобную для пользователя директорию на компьютере.

3.3.4.3 Выполнение программы

Для начала работы с приложением необходимо запустить исполняемый файл модуля-конфигуратора «HOTKEY BINDER.exe» в каталоге программы, откроется главное окно программы со списком горячих клавиш. Если программа уже запускалась ранее, то на экране могут быть отображены назначенные горячие клавиши. Если же приложение запускается в первый раз, то список будет пустым (рисунок А.1).

Чтобы добавить новую комбинацию клавиш, нужно в нижней части окна программы нажать на кнопку «Добавить». После этого в списке клавиш появится новая строка, состоящая из названия клавиши, наименования горячей клавиши и пути к исполняемому файлу программы, которую планируется запускать. По умолчанию клавишей-модификатором выбрана Ctrl, остальные поля будут пустыми (рисунок А.2). Для выбора клавиши-модификатора используйте выпадающий список в начале строки. Чтобы назначить горячую клавишу, кликните мышью на пустое поле для ввода справа от клавиши модификатора и знака «+». Поле поменяет цвет на красный и в нем отобразится сообщение «Нажмите клавишу» (рисунок А.3). Нажмите на клавиатуре клавишу, которую хотите использовать для быстрого доступа к приложению. После нажатия на клавишу поле для ввода снова поменяет цвет на светло-серый и в нем отобразится наименование нажатой вами клавиши. Если вы ошиблись при вводе клавиши, то ее можно поменять, если снова нажать на поле с названием горячей клавиши. Путь до исполняемого файла можно выбрать двумя способами: через диалоговое окно или вручную. Для вызова диалогового окна нажмите на кнопку с изображением трех точек «…» в конце строки. В открывшемся окне выберите исполняемый файл нужной программы или его ярлык, нажмите кнопку «Открыть». Диалоговое окно закроется и в строке слева от кнопки «…» появится путь до выбранной вами программы. Ручной способ заключается во вводе полного пути к файлу в текстовое поле и рекомендуется только для опытных пользователей.

Для сохранения внесенных изменений нажмите кнопку «Сохранить» в нижней части окна программы. Рядом отобразится сообщение об успешном сохранении конфигурации (рисунок А.4).

Если вы выполнили какие-то действия в программе, которые хотели бы отменить и еще не сохранили изменения, то нажав кнопку «Отмена» можно вернуться к последней сохраненной конфигурации.

Чтобы удалить какое-то приложение из списка, достаточно очистить соответствующую строку в столбце «Путь до программы» и сохранить конфигурацию. Пустые и несуществующие пути при сохранении исключаются из списка.

После настройки всех горячих клавиш требуется запустить модуль для считывания нажатий и запуска приложений. Для этого нажмите кнопку «Запуск» в нижней части окна программы. Рядом с кнопкой появится сообщение об успешном выполнении операции (рисунок А.5). Это действие необходимо выполнять после каждой процедуры настройки, чтобы изменения вступили в силу. В противном случае конфигурация будет применена только при следующем запуске системы.

Для завершения работы с модулем-конфигуратором нажмите «Выход» в нижней части программы или значок выхода «х» в верхнем правом углу.

При необходимости модуль-конфигуратор можно в любое время открыть повторно снова запустив «HOTKEY BINDER.exe» в корневом каталоге программы.

Если настройки новых горячих клавиш или изменения старых не требуется, то запуск конфигуратора больше не нужен. Программа будет запускаться самостоятельно при старте системы и работать в фоновом режиме.

Для запуска нужного приложения нажмите настроенную ранее комбинацию клавиш, приложение будет запущено автоматически.

# 3.4 Сравнительная характеристика работы

Аналоги программы, как AutoHotkey или Launchy, обладают большим функционал в сравнении с разработанной программой, но при этом имеют сложный перегруженный интерфейс. Простота интерфейса полученной при разработке курсового проекта делает его более привлекательным для обычного, не имеющего специальных знаний, пользователя.

# Заключение

В рамках данного курсового проекта была разработана программа для управления горячими клавишами, которая позволяет пользователю назначать комбинации клавиш для быстрого запуска приложений. Программа состоит из двух основных компонентов: программы конфигуратора с графическим интерфейсом для настройки горячих клавиш и основной программы, которая отслеживает нажатия клавиш с помощью системного хука.

Пользователь может легко добавлять, редактировать и удалять горячие клавиши через интуитивно понятный интерфейс, а реализация системного хука позволяет программе перехватывать нажатия клавиш даже вне окна приложения, что делает ее универсальной и удобной в использовании. Программа добавляется в автозагрузку, что обеспечивает ее постоянную работу в фоновом режиме. В программе предусмотрена проверка корректности вводимых данных, что повышает ее надежность.

Разработанная программа успешно решает поставленную задачу и демонстрирует применение современных технологий, таких как глобальные хук. Проект может быть использован как основа для более сложных решений, связанных с автоматизацией задач.

# Список литературы

1. Thai T. L., Lam H. . NET framework essentials. – " O'Reilly Media, Inc.", 2003.

2. Постолит А. В. Visual Studio. NET. – 2003.

3. Троелсен Э., Джепикс Ф. Язык программирования C# 7 и платформы. NET и. NET Core. – Litres, 2019.

4. Уоткинз Д., Хаммонд М., Эйбрамз Б. Программирование на платформе. NET. – 2003.

5. Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts // Taylor & Francis Online URL: https://www.tandfonline.com/doi/abs/10.1207/s15327590ijhc1802\_1 (дата обращения: 10.01.2025).

6. Hooks Overview // Microsoft Learn URL: https://learn.microsoft.com/en-us/windows/win32/winmsg/about-hooks (дата обращения: 14.01.2025).

7. Introduction to .NET // Microsoft Learn URL: https://learn.microsoft.com/en-us/dotnet/core/introduction (дата обращения: 10.01.2025).

8. Как написать кейлоггер на C# // spy-soft.net URL: https://spy-soft.net/keylogger-c/ (дата обращения: 12.01.2025).

9. Не «Ctrl+C»/«Ctrl+V» едиными. История клавиш-модификаторов // Хабр URL: https://habr.com/ru/companies/selectel/articles/839108/ (дата обращения: 11.01.2025).

10. Сочетания клавиш в Windows // Служба поддержки Майкрософт URL: https://support.microsoft.com/ru-ru/windows/%D1%81%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D1%8F-%D0%BA%D0%BB%D0%B0%D0%B2%D0%B8%D1%88-%D0%B2-windows-dcc61a57-8ff0-cffe-9796-cb9706c75eec#WindowsVersion=Windows\_10 (дата обращения: 10.01.2025).

# Приложение А (обязательное)

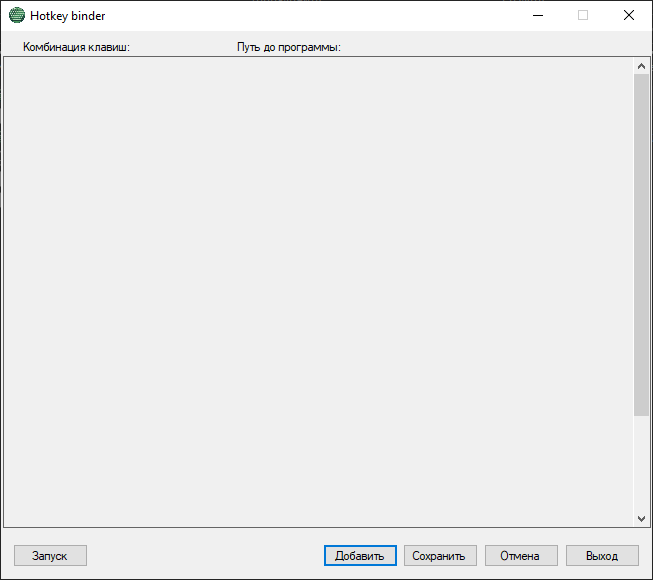


Рисунок А.1 — Первый запуск приложения

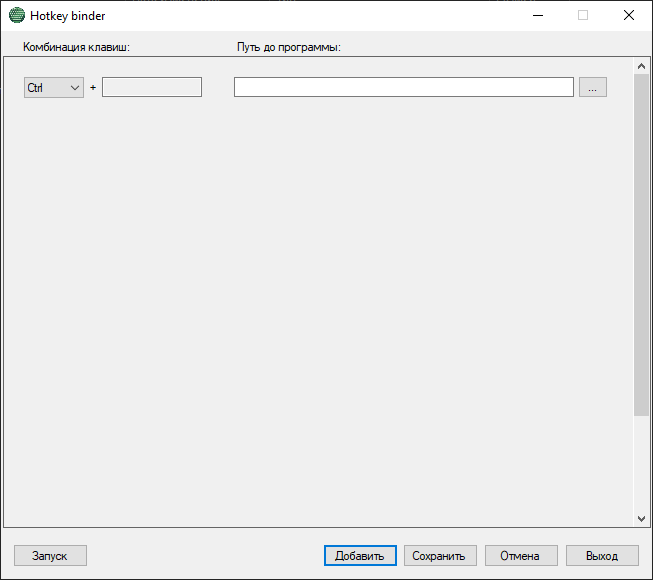


Рисунок А.2 — Добавление новой горячей клавиши

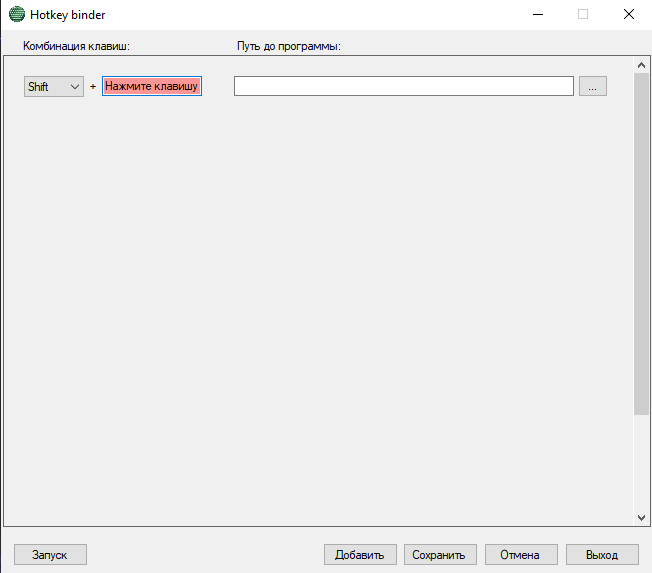


Рисунок А.3 — Запрос нажатия клавиши

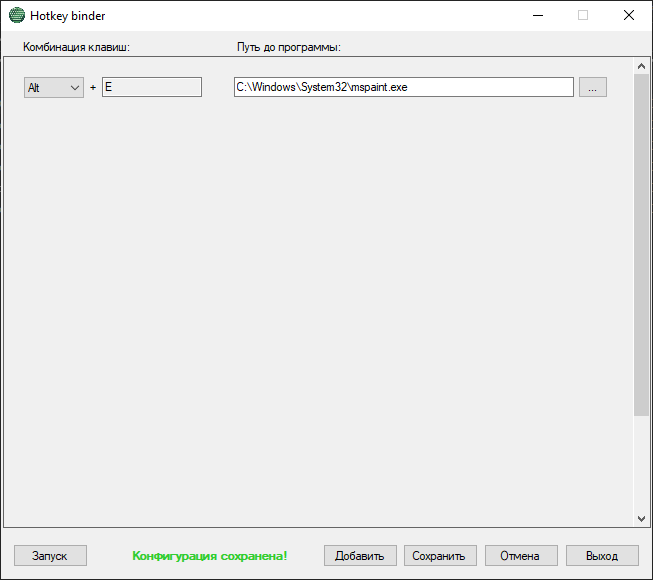


Рисунок А.4 — Сохранение конфигурации

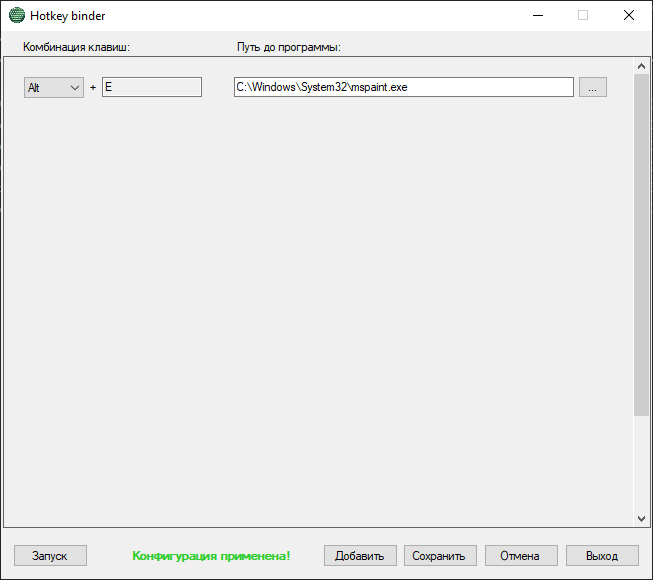


Рисунок А.5 — Применение конфигурации

# Приложение Б (обязательное)

Таблица Б.1 — Методы класса Hotkey\_binder

|  |  |
| --- | --- |
| Метод | Описание |
| Form1\_Load | Вызывается при загрузке формы, вызывает загрузчик конфигурации из файла load\_cfg и отображает интерфейс draw\_interface. |
| check\_cfg | Проверяет наличие конфигурационного файла hotkeys.cfg и создает его, если файл отсутствует. |
| load\_cfg | Проверяет наличие конфигурационного файла hotkeys.cfg и создает его, если файл отсутствует. |
| draw\_interface | Очищает список горячих клавиш hotkey\_list. Читает строки из hotkeys.cfg и парсит их с помощью регулярного выражения. Для каждой строки, прошедшей проверку, создает объект AppHotkey, который отображает модификатор, клавишу и путь к программе, которая будет запускаться. Некорректные клавиши подсвечиваются красным. |
| input\_key\_got\_focus | Вызывается при получении фокуса текстовым полем для ввода клавиши. Изменяет фон поля на красный и меняет текст на «Нажмите клавишу». |
| input\_key\_lost\_focus | Вызывается при получении фокуса текстовым полем для ввода клавиши. Восстанавливает предыдущее значение текстового поля и цвет фона. |
| input\_key\_key\_press | Вызывается при нажатии клавиши в текстовом поле. Меняет текст поля на наименование нажатой клавиши. |
| file\_path | Вызывается при нажатии кнопки «…». Открывает диалоговое окно выбора файла и записывает путь в соответствующее поле. |
| add\_string | Вызывается при нажатии кнопки «Добавить». Создает новый элемент AppHotkey, добавляет его на форму и обновляет интерфейс. |
| save\_config | Вызывается при нажатии на кнопку «Сохранить». Удаляет старый файл конфигурации и записывает новый на основе данных из списка hotkey\_list. |
| cancel\_click | Вызывается при нажатии кнопки «Отмена». Отменяет все изменения и откатывает интерфейс до состояния, соответствующего последней сохраненной конфигурации. |
| start\_main\_program | Запускает основное приложение Hotkey\_starter. Если приложение уже запущено, то перезапускает. |
| start\_program | Запускает процесс Hotkey\_starter.exe. |
| stop\_program | Завершает все запущенные процессы Hotkey\_starter.exe. |
| is\_autorun\_set | Проверяет наличие ярлыка основного приложения в автозагрузке. |
| set\_autorun | Создает ярлык в папке автозагрузки. |
| close\_program | Закрывает программу. |

# Приложение В (обязательное)

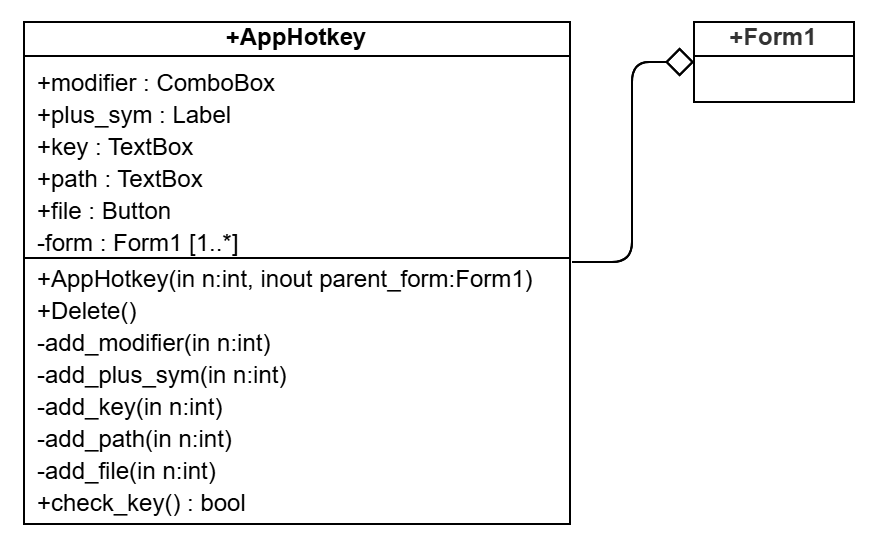


Рисунок В.1 — Диаграмма класса AppHotkey

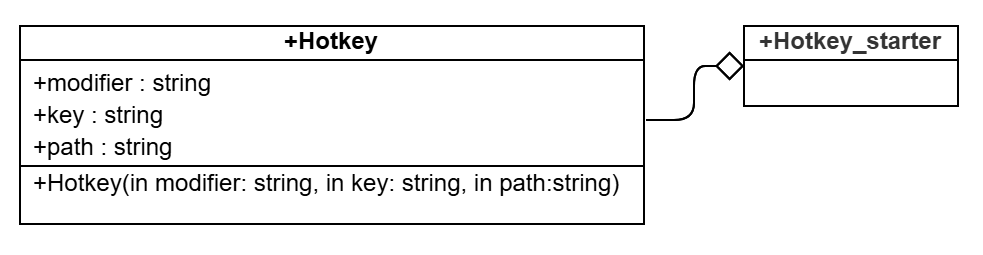


Рисунок В.2 — Диаграмма класса Hotkey

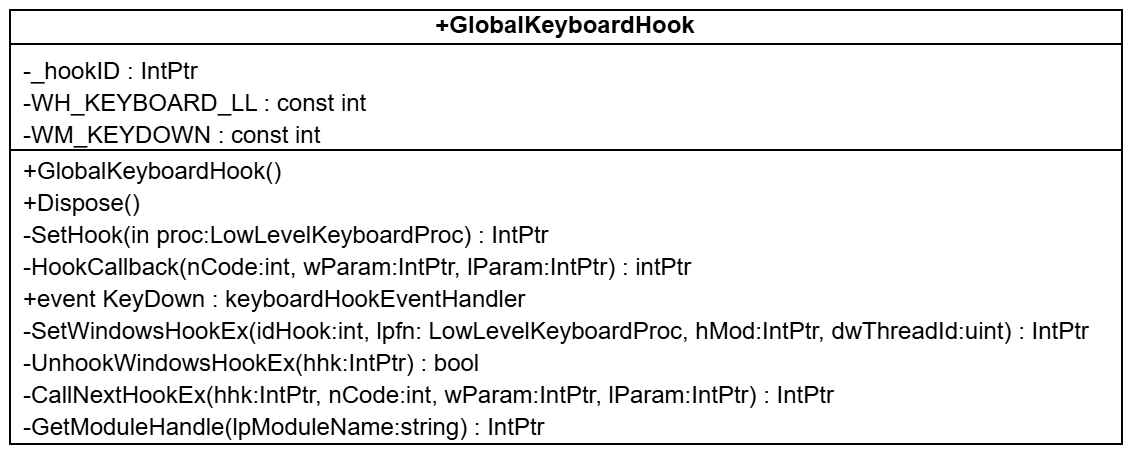


Рисунок В.3 — Диаграмма класса GlobalKeyboardHook

# Приложение Г (обязательное)



Рисунок Г.1 — Блок-схема метода Form1\_Load класса Form1



Рисунок Г.2 — Блок схема метода file\_path класса Form1



Рисунок Г.3 — Блок-схема метода input\_key\_got\_focus класса Form1



Рисунок Г.4 — Блок-схема метода input\_key\_lost\_focus класса Form1



Рисунок Г.5 — Блок-схема метода input\_key\_key\_press класса Form1



Рисунок Г.6 — Блок схема метода draw\_interface Form1



Рисунок Г.7 — Блок-схема метода check\_cfg класса Form1



Рисунок Г.8 — Блок схема метода load\_cfg класса Form1



Рисунок Г.9 — Блок-схема метода close\_program класса From1



Рисунок Г.10 — Блок-схема метода add\_string класса Form1



Рисунок Г.11 — Блок-схема метода save\_config класса Form1



Рисунок Г.12 — Блок-схема метода res\_focus\_lost класса Form1



Рисунок Г.13 — Блок-схема метода start\_main\_program класса Form1



Рисунок Г.14 — Блок схема метода cancel\_click класса Form1



Рисунок Г.15 — Блок схема метода start\_program класса Form1



Рисунок Г.16 — Блок-схема метода stop\_program класса Form1



Рисунок Г.17 — Блок-схема метода is\_autorun\_set класса Form1



Рисунок Г.18 — Блок-схема метода set\_autorun класса Form1



Рисунок Г.19 — Блок-схема конструктора класса AppHotkey



Рисунок Г.20 — Блок-схема метода Delete класса AppHotkey



Рисунок Г.21 — Блок-схема метода add\_modifier класса AppHotkey



Рисунок Г.22 — Блок-схема метода add\_plus\_sym класса AppHotkey



Рисунок Г.23 — Блок-схема метода add\_key класса AppHotkey



Рисунок Г.24 — Блок-схема метода add\_path класса AppHotkey



Рисунок Г.25 — Блок-схема метода add\_file класса AppHotkey



Рисунок Г.26 — Блок-схема метода check\_key класса AppHotkey



Рисунок Г.27 — Блок-схема метода Main класса Hotkey\_starter



Рисунок Г.28 — Блок-схема метода key\_down класса Hotkey\_starter



Рисунок Г.29 — Блок-схема метода get\_cfg класса Hotkey\_starter



Рисунок Г.30 — Блок-схема метода check\_cfg класса Hotkey\_starter



Рисунок Г.31 — Блок-схема метода open\_file класса Hotkey\_starter



Рисунок Г.32 — Блок-схема конструктора класса Hotkey



Рисунок Г.33 — Блок-схема конструктора класса GlobalKeyboardHook



Рисунок Г.34 — Блок-схема метода SetHook класса GlobalKeyboardHook



Рисунок Г.35 — Блок-схема метода HookCallback класса GlobalKeyboardHook



Рисунок Г.36 — Блок-схема метода Dispose класса GlobalKeyboardHook

# Приложение Д (обязательное)

Листинг Д.1 — Код файла Hotkey\_starter.cs

using System;

using System.Diagnostics;

using System.Windows.Forms;

using System.Text.RegularExpressions;

class Hotkey\_starter

{

private static readonly string maincfg = $"{Environment.CurrentDirectory}\\hotkeys.cfg";

private List<Hotkey> hotkeys = new List<Hotkey>();

public static void Main(string[] args)

{

Hotkey\_starter starter = new Hotkey\_starter();

starter.get\_cfg();

GlobalKeyboardHook keyboardHook = new GlobalKeyboardHook();

keyboardHook.KeyDown += starter.key\_down;

Application.Run();

}

private void key\_down(Keys key, bool ctrl, bool alt, bool shift)

{

String mod="";

if (ctrl) { mod = "Ctrl"; }

if (alt) { mod = "Alt"; }

if (shift) { mod = "Shift"; }

for (int i=0; i<hotkeys.Count; i++)

{

if ((hotkeys[i].modifier == mod) && (hotkeys[i].key==key.ToString()))

{

open\_file(hotkeys[i].path);

}

}

}

private void get\_cfg()

{

check\_cfg();

string[] lines = File.ReadAllLines(maincfg);

String pattern = @"^(?<modifier>Ctrl|Alt|Shift)\+(?<key>[A-Za-z0-9]+)=""(?<path>.\*)""$";

Match match;

Keys tmp\_key;

for (int i=0; i<lines.Length; i++)

{

match = Regex.Match(lines[i], pattern);

if ((match.Success) && (Enum.TryParse(match.Groups["key"].Value, out tmp\_key)))

{

hotkeys.Add(new Hotkey(match.Groups["modifier"].Value, match.Groups["key"].Value, match.Groups["path"].Value));

}

}

}

private void check\_cfg()

{

if (!File.Exists(maincfg))

{

File.Create(maincfg).Close();

}

}

private void open\_file(String path)

{

try

{

ProcessStartInfo processStartInfo = new ProcessStartInfo()

{

FileName = path,

};

Process process = Process.Start(processStartInfo);

}

catch (Exception ex)

{

}

}

}

Листинг Д.2 — Код файла GlobalKeyboardHook.cs

using System;

using System.Diagnostics;

using System.Runtime.InteropServices;

using System.Windows.Forms;

public class GlobalKeyboardHook : IDisposable

{

public delegate void KeyboardHookEventHandler(Keys key, bool ctrlPressed, bool altPressed, bool shiftPressed);

public event KeyboardHookEventHandler KeyDown;

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

private IntPtr \_hookID = IntPtr.Zero;

public GlobalKeyboardHook()

{

\_hookID = SetHook(HookCallback);

}

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

{

if (nCode >= 0 && wParam == (IntPtr)WM\_KEYDOWN)

{

int vkCode = Marshal.ReadInt32(lParam);

Keys key = (Keys)vkCode;

bool ctrlPressed = (Control.ModifierKeys & Keys.Control) == Keys.Control;

bool altPressed = (Control.ModifierKeys & Keys.Alt) == Keys.Alt;

bool shiftPressed = (Control.ModifierKeys & Keys.Shift) == Keys.Shift;

KeyDown?.Invoke(key, ctrlPressed, altPressed, shiftPressed);

}

return CallNextHookEx(\_hookID, nCode, wParam, lParam);

}

public void Dispose()

{

UnhookWindowsHookEx(\_hookID);

}

private const int WH\_KEYBOARD\_LL = 13;

private const int WM\_KEYDOWN = 0x0100;

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

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

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

[return: MarshalAs(UnmanagedType.Bool)]

private static extern bool UnhookWindowsHookEx(IntPtr hhk);

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

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

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

private static extern IntPtr GetModuleHandle(string lpModuleName);

}

Листинг Д.3 — Код файла Hotkey.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

public class Hotkey

{

public String modifier;

public String key;

public String path;

public Hotkey(string modifier, string key, string path)

{

this.modifier = modifier;

this.key = key;

this.path = path;

}

}

Листинг Д.4 — Код файла Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Text.RegularExpressions;

using System.Diagnostics;

using Microsoft.Win32;

using IWshRuntimeLibrary;

namespace Hotkey\_binder

{

public partial class Form1 : Form

{

public string[] form\_lines;

public List<AppHotkey> hotkey\_list = new List<AppHotkey>();

private string starter\_path = $"{Environment.CurrentDirectory}\\Hotkey\_starter.exe";

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

form\_lines = load\_cfg();

draw\_interface();

}

public void file\_path(object sender, EventArgs e)

{

Button btn = sender as Button;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

hotkey\_list[(int)btn.Tag].path.Text = openFileDialog1.FileName;

}

}

public void input\_key\_got\_focus(object sender, EventArgs e)

{

TextBox txt = sender as TextBox;

if ((txt.Tag == null) || (txt.Tag.ToString() != "not key")) { txt.Tag = txt.Text; }

txt.Text = "Нажмите клавишу";

txt.BackColor = Color.FromArgb(255, 150, 150);

}

public void input\_key\_lost\_focus(object sender, EventArgs e)

{

TextBox txt = sender as TextBox;

if (txt.Text == "Нажмите клавишу")

{

txt.Text = txt.Tag.ToString();

}

if (txt.Tag.ToString() != "not key")

{

txt.BackColor = SystemColors.Control;

}

}

public void input\_key\_key\_press(object sender, KeyEventArgs e)

{

TextBox txt = sender as TextBox;

txt.Text = e.KeyCode.ToString();

p1.Focus();

}

private void draw\_interface()

{

hotkey\_list.Clear();

String pattern = @"^(?<modifier>Ctrl|Alt|Shift)\+(?<key>[A-Za-z0-9]+)=""(?<path>.\*)""$";

Match match;

for (int i = 0; i < form\_lines.Length; i++)

{

match = Regex.Match(form\_lines[i], pattern);

if (match.Success)

{

hotkey\_list.Add(new AppHotkey(i, this));

hotkey\_list[i].modifier.Text = match.Groups["modifier"].Value;

hotkey\_list[i].key.Text = match.Groups["key"].Value;

if (!hotkey\_list[i].check\_key()) {

hotkey\_list[i].key.BackColor = Color.FromArgb(255, 150, 150);

hotkey\_list[i].key.Tag = "not key";

}

hotkey\_list[i].path.Text = match.Groups["path"].Value;

}

}

}

public void check\_cfg()

{

if (!System.IO.File.Exists(Program.maincfg))

{

System.IO.File.Create(Program.maincfg).Close();

}

}

private string[] load\_cfg()

{

check\_cfg();

string[] lines = System.IO.File.ReadAllLines(Program.maincfg);

return lines;

}

private void close\_program(object sender, EventArgs e) //Закрыть

{

this.Close();

}

private void add\_string(object sender, EventArgs e) //Добавить

{

int list\_size = hotkey\_list.Count;

int curScrPos = p1.VerticalScroll.Value;

p1.VerticalScroll.Value = 0;

p1.Invalidate();

hotkey\_list.Add(new AppHotkey(list\_size, this));

p1.Invalidate();

if (list\_size < 16)

{

p1.VerticalScroll.Value = curScrPos;

}

else

{

p1.VerticalScroll.Value = p1.VerticalScroll.Maximum;

}

p1.PerformLayout();

}

private void save\_config(object sender, EventArgs e) //Сохранить

{

System.IO.File.Delete(Program.maincfg);

AppHotkey tmp;

for (int i = 0; i < hotkey\_list.Count; i++)

{

tmp = hotkey\_list[i];

if ((tmp.check\_key()) && (System.IO.File.Exists(tmp.path.Text)))

{

System.IO.File.AppendAllText(Program.maincfg, tmp.modifier.Text + "+" + tmp.key.Text + "=\"" + tmp.path.Text + "\"\r\n");

}

}

cancel\_click(sender, e);

res\_lbl.Text = "Конфигурация сохранена!";

res\_lbl.Visible = true;

res\_lbl.Focus();

}

private void res\_focus\_lost(object sender, EventArgs e)

{

res\_lbl.Visible = false;

}

private void start\_main\_program(object sender, EventArgs e) //Запустить

{

if (is\_autorun\_set())

{

stop\_program();

} else

{

set\_autorun();

}

start\_program();

res\_lbl.Text = "Конфигурация применена!";

res\_lbl.Visible = true;

res\_lbl.Focus();

}

private void cancel\_click(object sender, EventArgs e) //Отмена

{

for (int i = 0; i < hotkey\_list.Count; i++)

{

hotkey\_list[i].Delete();

}

form\_lines = load\_cfg();

draw\_interface();

}

private void start\_program()

{

Process proc = new Process();

proc.StartInfo.FileName = starter\_path;

proc.Start();

}

private void stop\_program()

{

Process[] process\_list = Process.GetProcessesByName("Hotkey\_starter");

if (process\_list.Length>0)

{

foreach (Process process in process\_list)

{

process.Kill();

}

}

}

private bool is\_autorun\_set()

{

return System.IO.File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.Startup) + "\\Hotkey Starter.lnk");

}

private void set\_autorun()

{

string autorun\_link = Environment.GetFolderPath(Environment.SpecialFolder.Startup) + "\\Hotkey Starter.lnk";

WshShell wshShell = new WshShell();

IWshShortcut shortcut = (IWshShortcut)wshShell.CreateShortcut(autorun\_link);

shortcut.TargetPath = starter\_path;

shortcut.WorkingDirectory = Path.GetDirectoryName(starter\_path);

shortcut.IconLocation = Path.GetDirectoryName(starter\_path)+"\\icon.ico";

shortcut.Save();

}

}

}

Листинг Д.5 — Код файла AppHotkey.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace Hotkey\_binder

{

public class AppHotkey

{

public ComboBox modifier;

public Label plus\_sym;

public TextBox key, path;

public Button file;

private Form1 form;

public AppHotkey(int n, Form1 parent\_form)

{

form = parent\_form;

add\_modifier(n);

add\_plus\_sym(n);

add\_key(n);

add\_path(n);

add\_file(n);

}

public void Delete()

{

form.p1.Controls.Remove(modifier);

modifier.Dispose();

form.p1.Controls.Remove(plus\_sym);

plus\_sym.Dispose();

form.p1.Controls.Remove(key);

key.Dispose();

form.p1.Controls.Remove(path);

path.Dispose();

form.p1.Controls.Remove(file);

file.Dispose();

}

private void add\_modifier(int n)

{

modifier = new ComboBox();

modifier.DropDownStyle = ComboBoxStyle.DropDownList;

modifier.Width = 60;

modifier.Items.Add("Ctrl");

modifier.Items.Add("Shift");

modifier.Items.Add("Alt");

modifier.SelectedIndex = 0;

modifier.Location = new Point(20, 20 + n \* 26);

modifier.Visible = true;

form.p1.Controls.Add(modifier);

}

private void add\_plus\_sym(int n)

{

plus\_sym = new Label();

plus\_sym.Text = "+";

plus\_sym.Width = 10;

plus\_sym.Location = new Point(83, 23 + n \* 26);

plus\_sym.Visible = true;

form.p1.Controls.Add(plus\_sym);

}

private void add\_key(int n)

{

key = new TextBox();

key.Location = new Point(98, 20 + n \* 26);

key.Width = 100;

key.ReadOnly = true;

key.GotFocus += form.input\_key\_got\_focus;

key.LostFocus += form.input\_key\_lost\_focus;

key.KeyDown += form.input\_key\_key\_press;

key.Visible = true;

form.p1.Controls.Add(key);

}

private void add\_path(int n)

{

path = new TextBox();

path.Location = new Point(230, 20 + n \* 26);

path.Width = 340;

path.Visible = true;

form.p1.Controls.Add(path);

}

private void add\_file(int n)

{

file = new Button();

file.Tag = n;

file.Location = new Point(574, 19 + n \* 26);

file.Width = 30;

file.Height = 22;

file.Text = "...";

file.Click += form.file\_path;

file.Visible = true;

form.p1.Controls.Add(file);

}

public bool check\_key()

{

Keys key\_inside;

if (Enum.TryParse(key.Text, out key\_inside))

{

return true;

} else

{

return false;

}

}

}

}