*Министерство науки и высшего образования Российской федерации*

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

РФ КГУ 09.03.04КР23 375226

Кафедра «Программное обеспечение автоматизированных систем»

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Операционные системы»**

**на тему: «Разработка службы Windows на языке C#»**

Выполнил студент гр. ИТ-0940321 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Колягин А.А./

подпись, дата

Направление 09.03.04 - «Программная инженерия»

Проверил канд. физ.- мат. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Рабушко А.Г./

подпись, дата

Работа защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2023

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc154596244)

[ВВЕДЕНИЕ 3](#_Toc154596245)

[РЕАЛИЗАЦИЯ 4](#_Toc154596246)

[Средство реализации 4](#_Toc154596247)

[Пошаговая инструкция как создать службу на С# 5](#_Toc154596248)

[Удаление службы 20](#_Toc154596249)

[ЗАКЛЮЧЕНИЕ 21](#_Toc154596250)

[СПИСОК ЛИТЕРАТУРЫ 22](#_Toc154596251)

# ВВЕДЕНИЕ

Службы Windows играют ключевую роль в обеспечении стабильности и функциональности операционной системы. Эти фоновые процессы выполняют разнообразные задачи, такие как обработка данных, мониторинг и обеспечение безопасности.

Разработка собственной службы на языке C# предоставляет уникальную возможность создания индивидуальных фоновых приложений под управлением Windows. Это особенно актуально в контексте популярности операционной системы Windows, где службы играют важную роль в обеспечении бесперебойной работы приложений.

# РЕАЛИЗАЦИЯ

## Средство реализации

Для разработки собственной службы Windows на языке C# в среде Visual Studio 2022, используются современные инструменты и библиотеки, обеспечивающие эффективное создание и управление службой. Такой выбор был сделан из-за развития и популярности языка C# в мире программирования, а также его мощной экосистемы разработки и инструментов поддержки. C# предоставляет разнообразные возможности для создания служб и позволяет легко взаимодействовать с другими компонентами системы.

1. Среда разработки Visual Studio 2022:
   * Интегрированная среда разработки от Microsoft обеспечивает удобный интерфейс для создания проектов на C#.
   * Visual Studio 2022 предоставляет широкие возможности по отладке, анализу кода и автоматической сборке проекта.
2. Язык программирования C#:
   * Используется для написания основного кода службы, что обеспечивает высокую производительность и простоту разработки.
3. Библиотека .NET:
   * .NET Framework или .NET Core предоставляют набор библиотек для работы с службами Windows.
   * Возможности .NET включают в себя классы для создания, управления и взаимодействия с службами.

## Пошаговая инструкция как создать службу на С#

Шаг 1: Создание проекта службы

1. Откройте Visual Studio 2022 и создайте новый проект.
2. В разделе "Поиск шаблонов" выберите "Служба Windows (.NET Core)" или "Служба Windows (.NET Framework)" в зависимости от вашего предпочтения.
3. Настройте параметры проекта, включая имя, расположение и фреймворк. Нажмите "Создать".

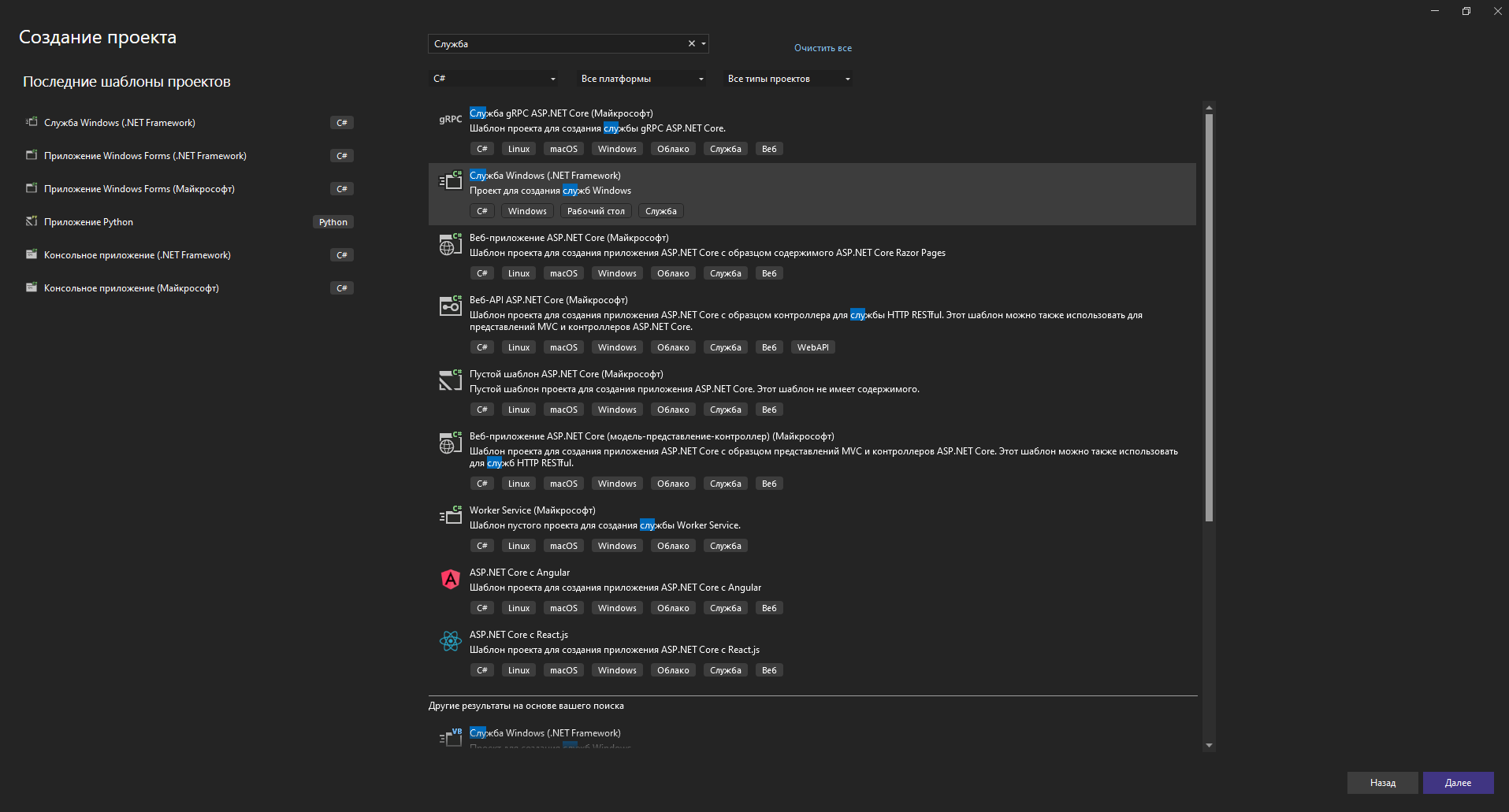


Рисунок 1 – Создание проекта

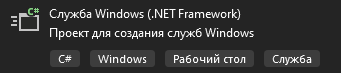


Рисунок 2 – Выбор шаблона проекта

Выбираем службу Windows (.Net Framework) именно на С#. Далее мы настраиваем новый проект и называем его, у нас будет название MyService(Рис 3).

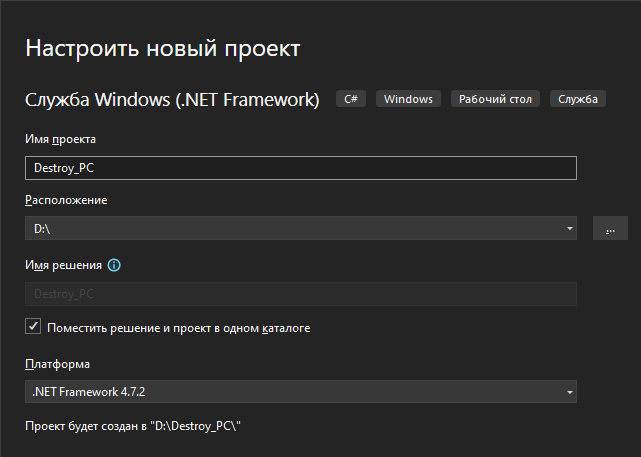


Рисунок 3 – Настройка проекта

Настраиваем новый проект и нажимаем “Создать” Далее у нас создается вот такой проект как на рисунке 4

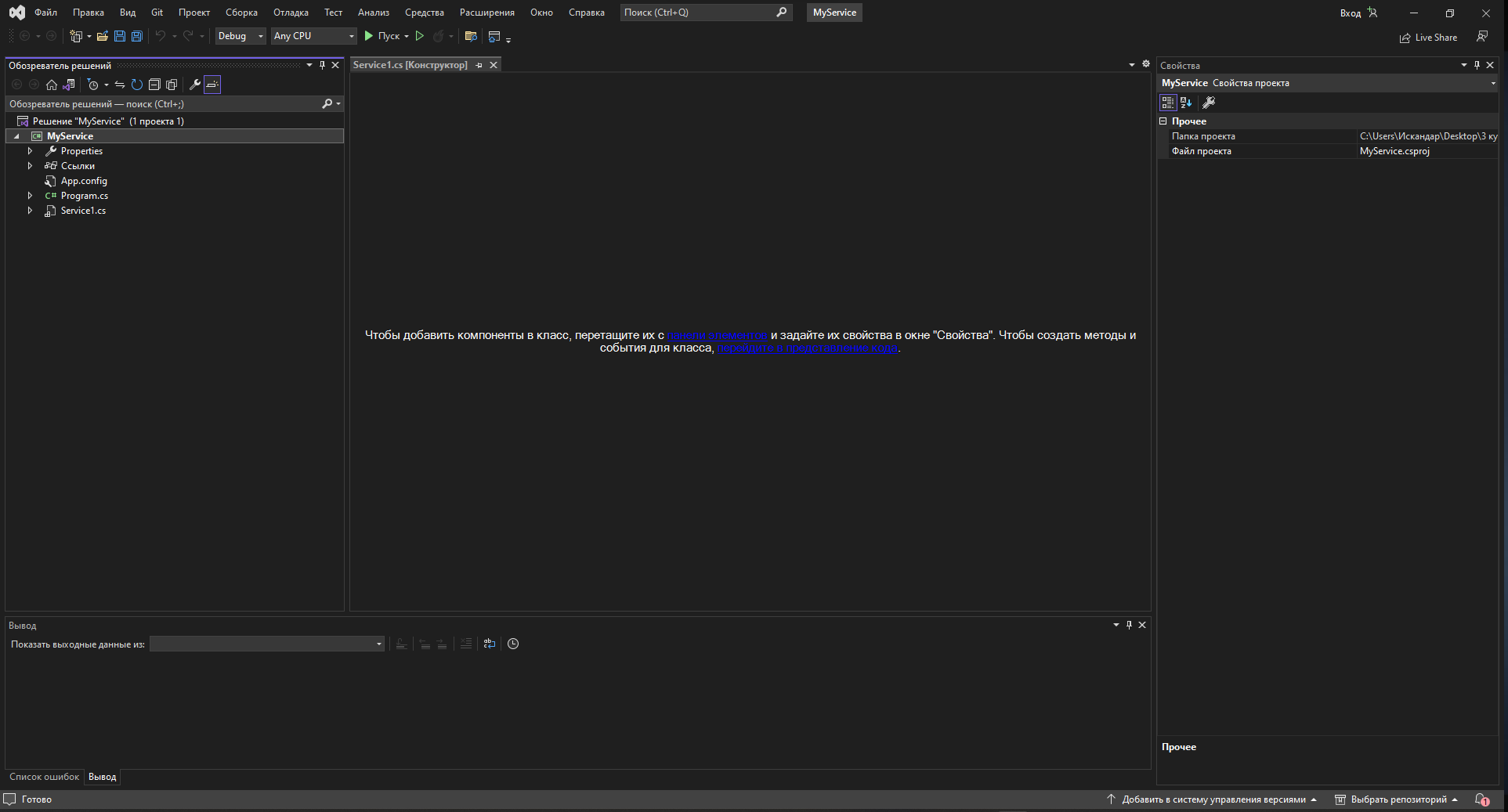


Рисунок 4 – Созданный проект

Шаг 2: Написание кода службы

Дальше мы обязательно должны добавить установщик это делается так, нажимаем левую кнопку мыши и выбираем “Добавить установщик”.

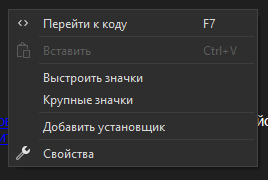


Рисунок 5 – Добавление установщика

У нас создается такой файл как “ProjectInstaller.cs” в котором есть два эллемента: serviceProcessInstaller1

1. ServiceProcessInstaller1
2. ServiceInstaller1

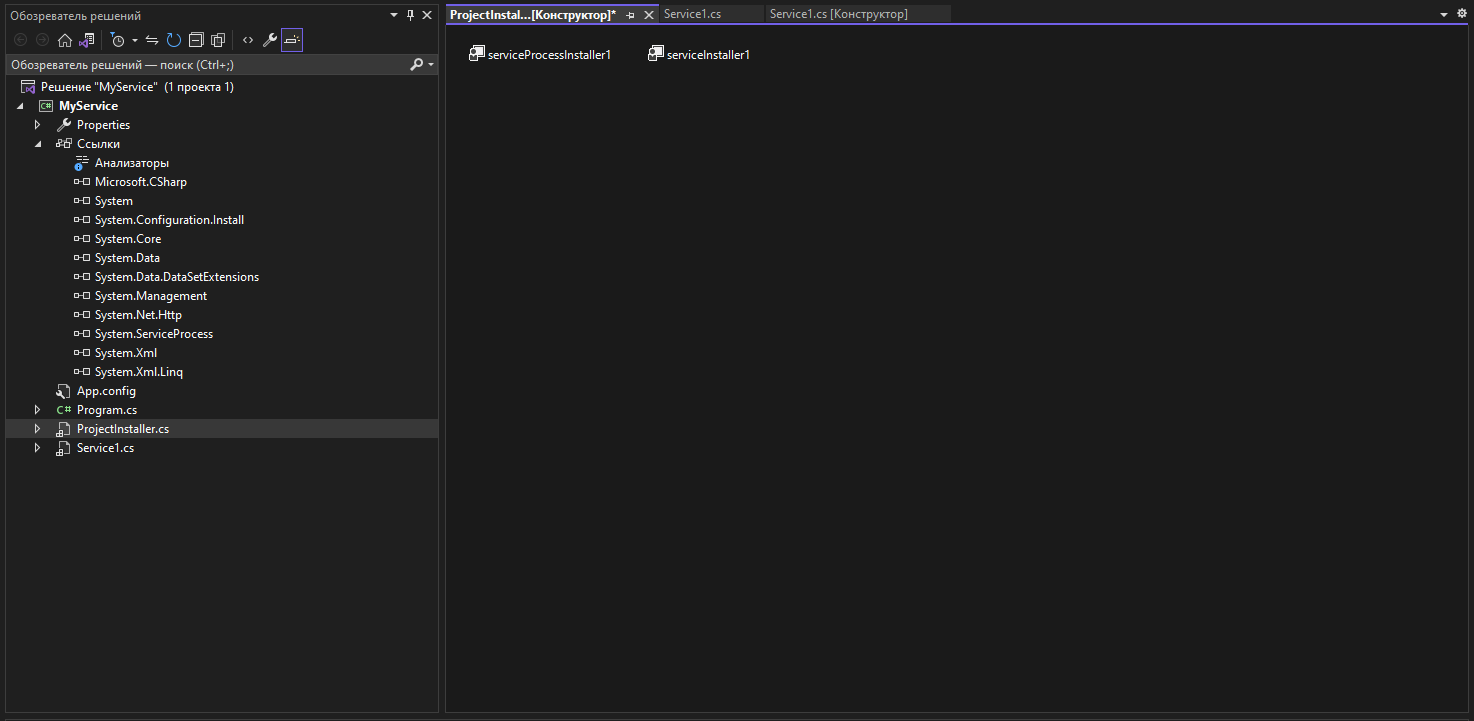


Рисунок 6 – Автоматически созданные элементы

ServiceProcessInstaller1:

Описание: Этот компонент представляет инсталлятор процесса службы, который определяет, от имени какого пользователя будет запущена служба, и какие привилегии у нее будут.

Свойства:

* Account: Определяет, от имени какого пользователя будет выполняться служба. Обычно используются значения LocalSystem, LocalService или NetworkService.
* Password: Задает пароль для учетной записи, если Account установлен в User.

ServiceInstaller1:

Описание: Этот компонент представляет инсталлятор службы и определяет базовые параметры службы, такие как ее имя и отображаемое имя (DisplayName).

Свойства:

* ServiceName: Уникальное имя службы, используемое для управления ею через командную строку или API.
* DisplayName: Отображаемое имя службы, которое видно в службах Windows.
* StartType: Определяет, как служба будет запущена по умолчанию (Automatic, Manual, Disabled).

Эти компоненты обеспечивают конфигурацию и установку вашей службы в операционной системе. Важно правильно настроить параметры ServiceProcessInstaller1 и ServiceInstaller1 в соответствии с требованиями вашей службы и окружения, в котором она будет работать.

Далее мы настраиваем свойства созданных эллементов (Рис 7).

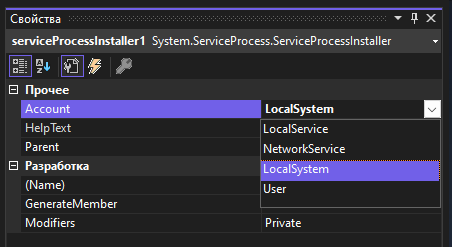


Рисунок 7 – Свойства ServiceProcessInstaller1

Мы выбираем из свойств Account более нам подходящую :

1. LocalSystem (Локальная система):
   * Описание: Служба будет запущена от имени локальной системной учетной записи.
   * Применение: Используется, когда служба не требует доступа к сетевым ресурсам.
2. LocalService (Локальная служба):
   * Описание: Служба будет запущена от имени локальной службенной учетной записи.
   * Применение: Подходит для служб, которые не требуют расширенных привилегий.
3. NetworkService (Сетевая служба):
   * Описание: Служба будет запущена от имени сетевой службенной учетной записи.
   * Применение: Рекомендуется, если служба требует доступа к сетевым ресурсам.
4. User (Пользователь):
   * Описание: Служба будет запущена от имени указанной пользовательской учетной записи.
   * Применение: Используется, когда служба требует определенных прав или доступа к ресурсам, доступным только конкретному пользователю.

Нам подходит LocalSystem потому что выполняется с полными привилегиями администратора, имеет доступ к практически всем ресурсам и может выполнять системные операции на высоком уровне.

Дальше мы меняем свойства в элементе ServiceInstaller1(Рис 8).

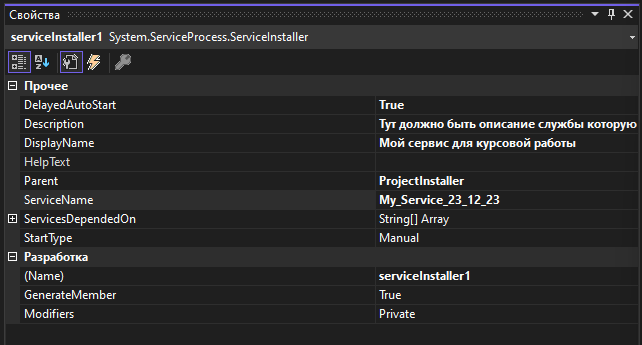


Рисунок 8 – Свойства ServiceInstaller1

Свойства объекта ServiceInstaller1 включают:

1. **ServiceName (Имя службы):**
   * **Описание:** Уникальное имя службы, используемое для идентификации ее в системе.
   * **Применение:** Используется при установке, удалении и управлении службой через командную строку или API.
2. **DisplayName (Отображаемое имя):**
   * **Описание:** Имя службы, отображаемое в окне "Службы Windows" для удобства пользователя.
   * **Применение:** Улучшает читаемость и идентификацию службы в пользовательском интерфейсе операционной системы.
3. **Description (Описание):**
   * **Описание:** Текстовое описание службы, предоставляющее пользователю дополнительную информацию о ее назначении.
   * **Применение:** Повышает понимание функций и роли службы.
4. **StartType (Тип запуска):**
   * **Описание:** Определяет, как служба будет запущена по умолчанию при старте операционной системы.
   * **Применение:** Возможные значения включают Automatic (Автоматически), Manual (Вручную) и Disabled (Отключено).
5. **DelayedAutoStart (Отложенный автозапуск):**
   * **Описание:** Указывает, следует ли отложить запуск службы автоматически после запуска системы.
   * **Применение:** Используется, чтобы уменьшить загрузку системы при старте.
6. **Context (Контекст):**
   * **Описание:** Определяет контекст установки, используемый для конфигурации параметров службы во время установки.
   * **Применение:** Предоставляет дополнительные настройки, связанные с установкой службы.
7. **Parent (Родитель):**
   * **Описание:** Ссылка на родительский объект, к которому привязан инсталлятор службы.
   * **Применение:** Установка связей между инсталляторами службы и другими компонентами проекта.

Далее откройте файл Service1.cs (или аналогичный) в редакторе кода перейдите в “представление кода ” (Рис. 9).

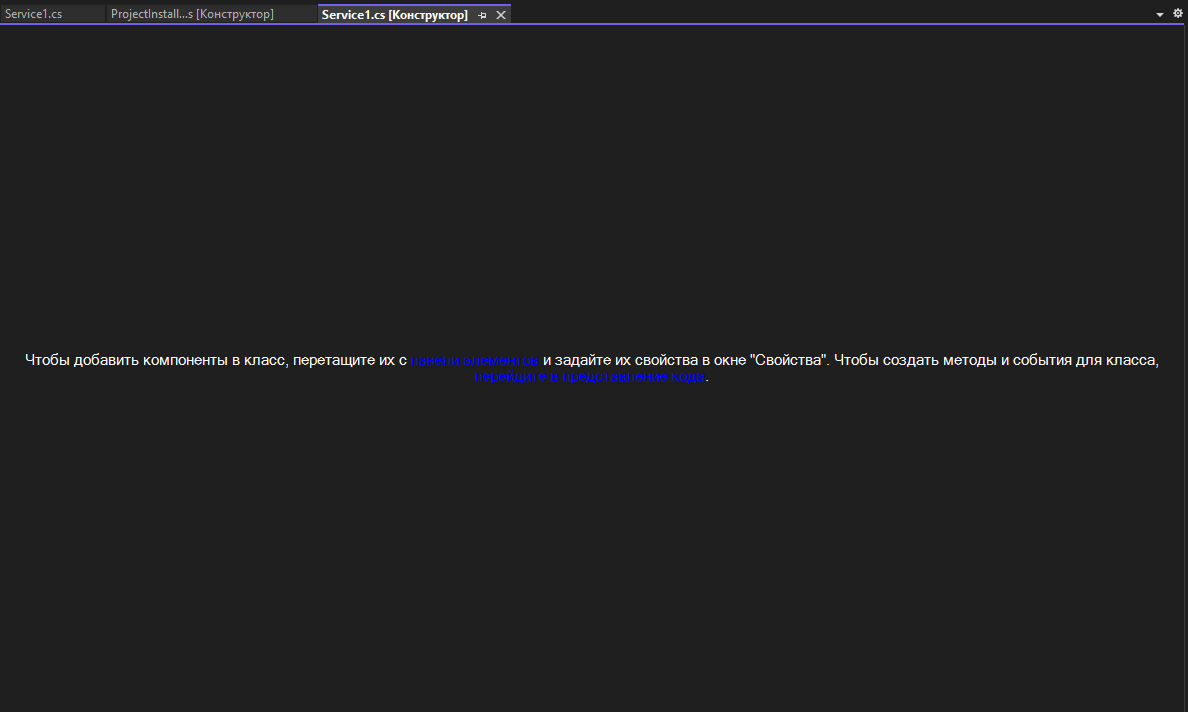


Рисунок 9 –Переход к коду проекта в Service1.cs

Реализуйте свою логику в методах OnStart и OnStop, которые вызываются при запуске и остановке службы соответственно.

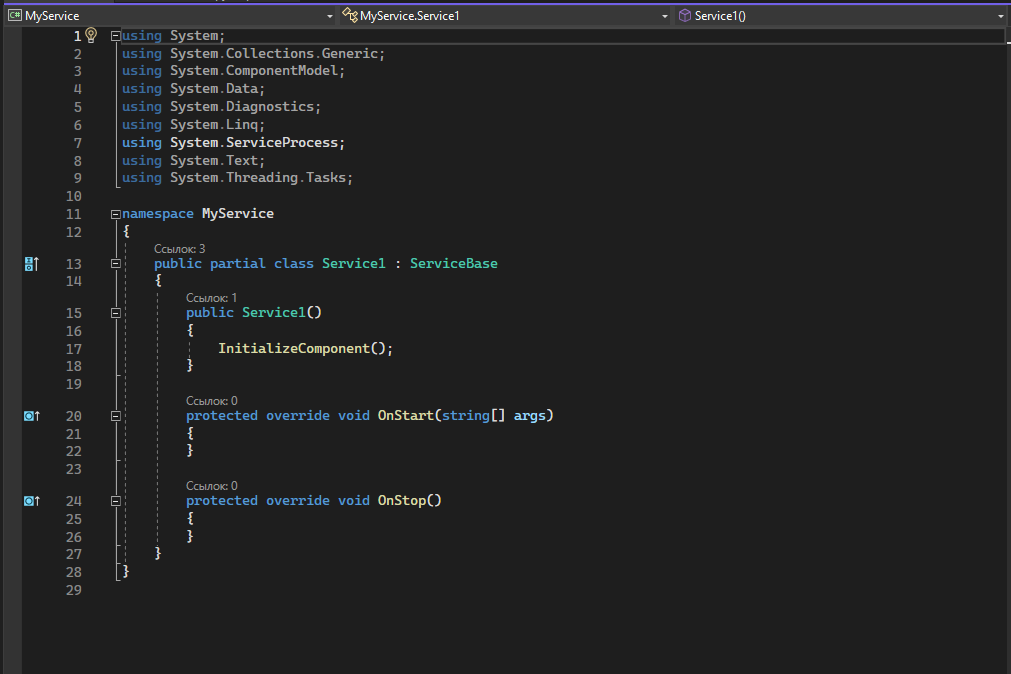


Рисунок 10 – Код программы созданный автоматически

Создадим службу Windows, которая мониторит указанный каталог на изменения в файлах. Давайте посмотри листинг кода:  
public partial class Service1 : ServiceBase

{

private FileSystemWatcher fileWatcher;

private string lastRenamedMessage;

public Service1()

{

InitializeComponent();

lastRenamedMessage = null;

}

protected override async void OnStart(string[] args)

{

// Задаем путь к каталогу для мониторинга

string directoryPath = @"D:\wiser";

// Инициализация FileSystemWatcher

fileWatcher = new FileSystemWatcher();

fileWatcher.Path = directoryPath;

fileWatcher.Filter = "\*.\*"; // Мониторим все файлы

// Подписываемся на события изменения файлов

fileWatcher.Created += OnFileChanged;

fileWatcher.Deleted += OnFileChanged;

fileWatcher.Changed += OnFileChanged;

fileWatcher.Renamed += OnFileRenamed;

// Включаем мониторинг

fileWatcher.EnableRaisingEvents = true;

string logMessage = $"Мониторинг каталога {directoryPath} запущен.";

EventLog.WriteEntry("ServiceKursovaya", logMessage);

AppendLogToFile(logMessage);

}

protected override async void OnStop()

{

string logMessage = "Служба остановлена.";

EventLog.WriteEntry("ServiceKursovaya", logMessage);

AppendLogToFile(logMessage);

// Выключаем мониторинг

fileWatcher.EnableRaisingEvents = false;

}

private void OnFileChanged(object sender, FileSystemEventArgs e)

{

// Код, который будет выполняться при создании, удалении или изменении файла

string changeType = e.ChangeType.ToString();

string fileName = Path.GetFileNameWithoutExtension(e.Name);

string logMessage = $"{changeType}: Файл \"{fileName}\": Log в каталоге {fileWatcher.Path}";

// Проверяем, было ли уже отображено сообщение о переименовании

if (lastRenamedMessage != logMessage && !logMessage.Contains("Changed"))

{

EventLog.WriteEntry("ServiceKursovaya", logMessage);

AppendLogToFile(logMessage);

lastRenamedMessage = logMessage;

}

else

{

// Если сообщение о переименовании уже было отображено или содержит "Changed", устанавливаем lastRenamedMessage в null

lastRenamedMessage = null;

}

}

private void OnFileRenamed(object sender, RenamedEventArgs e)

{

// Код, который будет выполняться при переименовании файла

string oldFileName = Path.GetFileNameWithoutExtension(e.OldName);

string newFileName = Path.GetFileNameWithoutExtension(e.Name);

string logMessage = $"Файл переименован: {oldFileName} => {newFileName}";

// Проверяем, было ли уже отображено сообщение о переименовании

if (lastRenamedMessage != logMessage)

{

EventLog.WriteEntry("ServiceKursovaya", logMessage);

AppendLogToFile(logMessage);

lastRenamedMessage = logMessage;

}

else

{

// Если сообщение о переименовании уже было отображено, устанавливаем lastRenamedMessage в null

lastRenamedMessage = null;

}

}

private void AppendLogToFile(string logMessage)

{

string logFilePath = "D:\\wiser\\Log.txt";

File.AppendAllText(logFilePath, $"{DateTime.Now}: {logMessage}\n");

}

}

Давайте разберем основные элементы кода:

**Служба мониторинга файлов:** Код создает службу Windows (Service1), которая отслеживает события в указанном каталоге и записывает их в журнал событий и в текстовый файл лога.

**Классы и методы:**

1. **Service1 (наследуется от ServiceBase):**
   * **fileWatcher (тип FileSystemWatcher):** Этот объект отвечает за отслеживание изменений в файловой системе.
   * **lastRenamedMessage (тип string):** Хранит последнее сообщение о переименовании файла для избежания дублирования записей.
2. **Service1() (конструктор):**
   * Инициализирует компоненты службы и устанавливает lastRenamedMessage в null.
3. **OnStart(string[] args):**
   * **Настройка FileSystemWatcher:** Устанавливает путь к каталогу для мониторинга, фильтр файлов (.), и подписывается на события создания, удаления, изменения и переименования файлов.
   * **Включение мониторинга:** Включает мониторинг файловой системы.
   * **Запись в журнал и лог-файл:** Записывает в журнал событий и текстовый лог сообщение о запуске мониторинга.
4. OnStop():
   * **Завершение службы:** Записывает в журнал и лог сообщение об остановке службы и выключает мониторинг файловой системы.
5. OnFileChanged(object sender, FileSystemEventArgs e):
   * **Обработка событий изменения файлов:** Генерирует сообщение о создании, удалении или изменении файла и записывает его в журнал и лог-файл.
6. OnFileRenamed(object sender, RenamedEventArgs e):
   * **Обработка события переименования файла:** Генерирует сообщение о переименовании файла и записывает его в журнал и лог-файл, предотвращая дублирование сообщений.
7. AppendLogToFile(string logMessage):
   * **Добавление записи в лог-файл:** Добавляет текущее время и сообщение в текстовый лог-файл.

Давайте дальше поменяем иконку службы, для этого открываем “Свойства ServiceKursovaya” в верхней вкладке “Проект” и далее выбираем заранее скачанную иконку для проекта с расширением ico.(Рис 11-12)

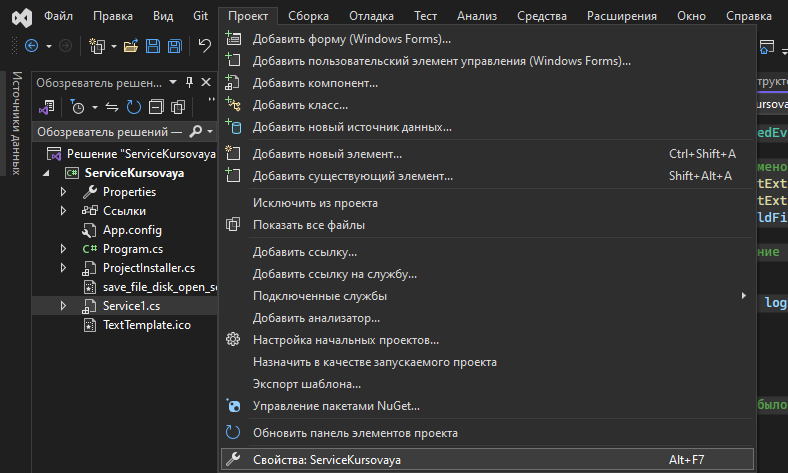


Рисунок 11 – Переход в свойства ServiceKursovaya

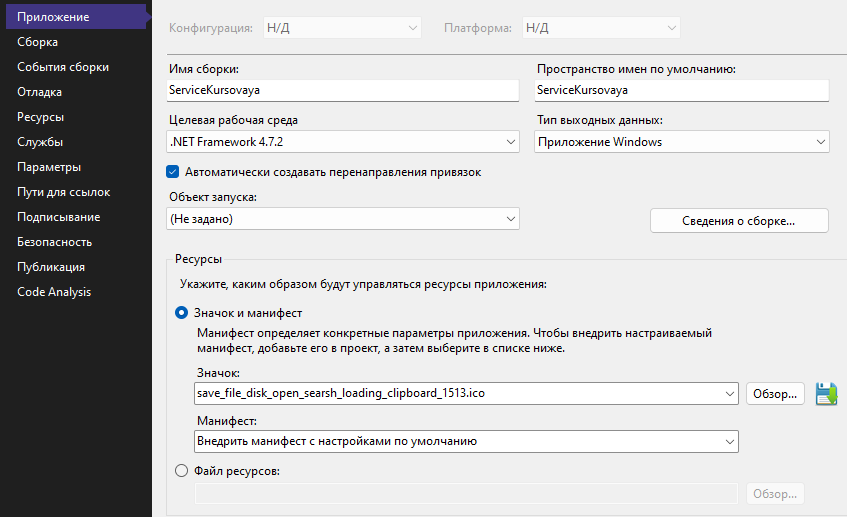


Рисунок 12 – Смена иконки

А теперь пробуем запустить проект и видим ошибку :

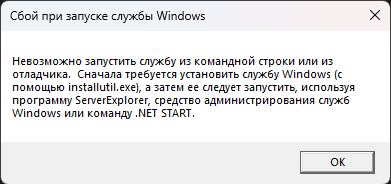


Рисунок 13 – Ошибка запуска службы

**Шаг 3: Установка и запуск службы**

Чтобы исправить эту ошибку нам нужно правильно запустить свою службу для начала мы находим “InstallUtil.exe” перейдя по этому пути:



Рисунок 14 - Расположение “InstallUtil.exe”

Следующим этапом открываем терминал от имени Администратора (cmd), переходим в папку указывая данную команду:  
 cd "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\” (рисунке 14)  
находим в этой папке файл InstallUtil.exe и перетаскиваем, либо указываем его путь в терминале, и добавляем путь нашего exe файла проекта(если его нет, соберите проект), должен получиться вот такой код:  
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe D:\ServiceKursovaya\bin\Debug\ServiceKursovaya.exe

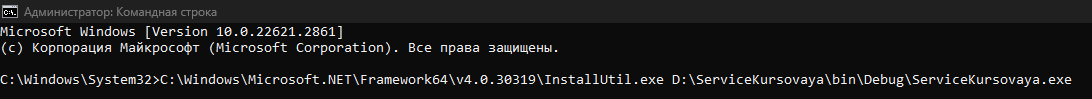


Рисунок 15 - Открыть в Терминале

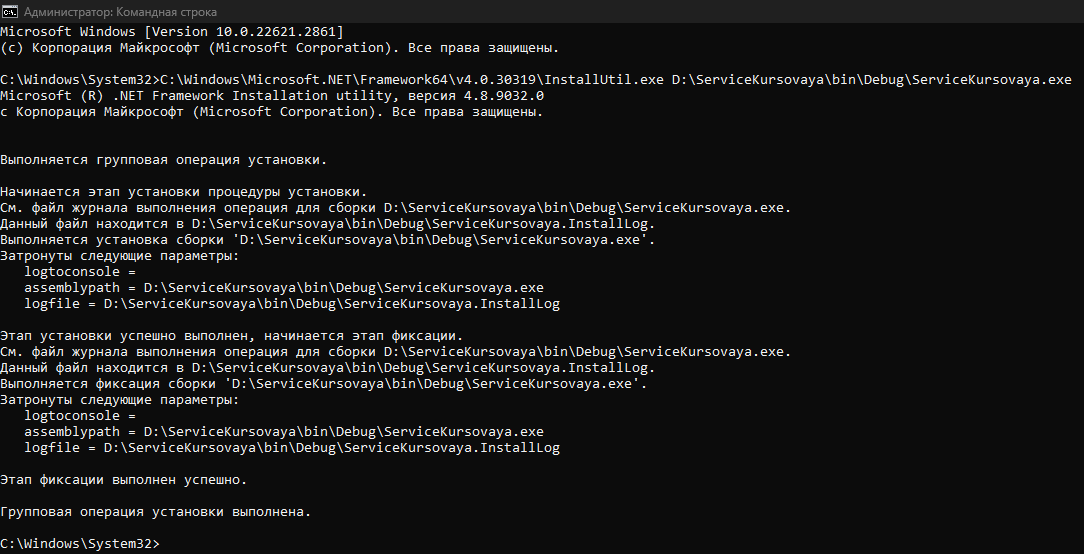


Рисунок 16 – Запуск через терминал

Чтобы посмотреть службы:

Щелкните правой кнопкой мыши кнопку «Пуск» и выберите «Управление компьютером» в контекстном меню»

В новом окне разверните «Службы и приложения» и нажмите на появившийся пункт «Службы»

Теперь справа появится список всех существующих служб в Windows 11. Заходим и видим свою службу

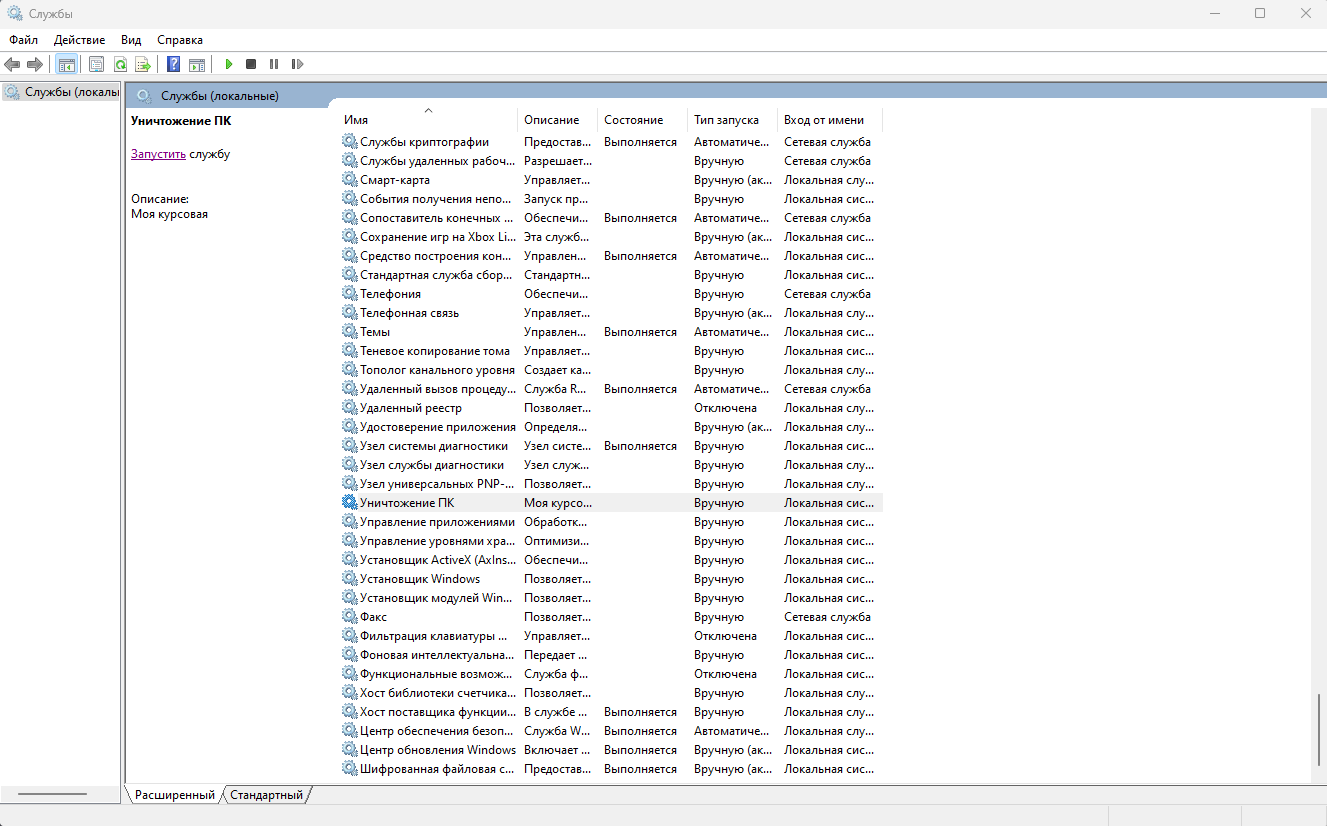


Рисунок 17 – Запуск службы

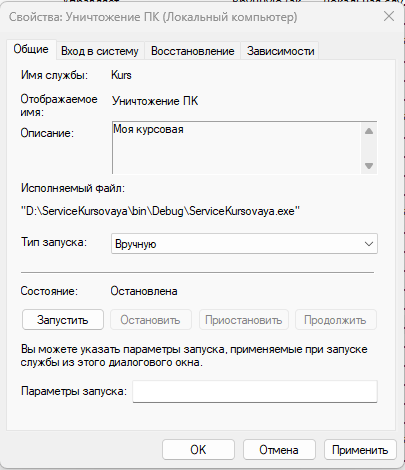


Рисунок 18 – Свойства созданной службы

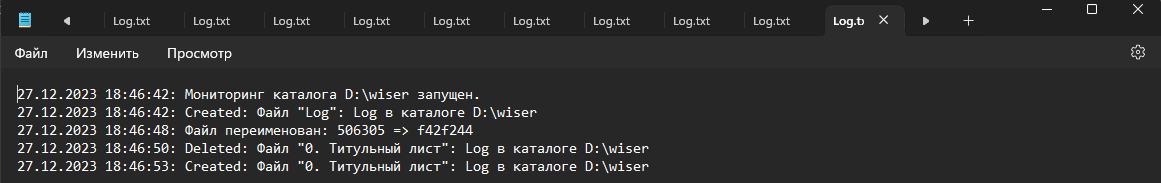


Рисунок 19 – Работа созданной службы

## Удаление службы

Для удаления службы также переходим в папку используем ту же команду но добавляем команду /u, которая использует на деинсталляцию службы:

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe D:\ServiceKursovaya\bin\Debug\ServiceKursovaya.exe /u

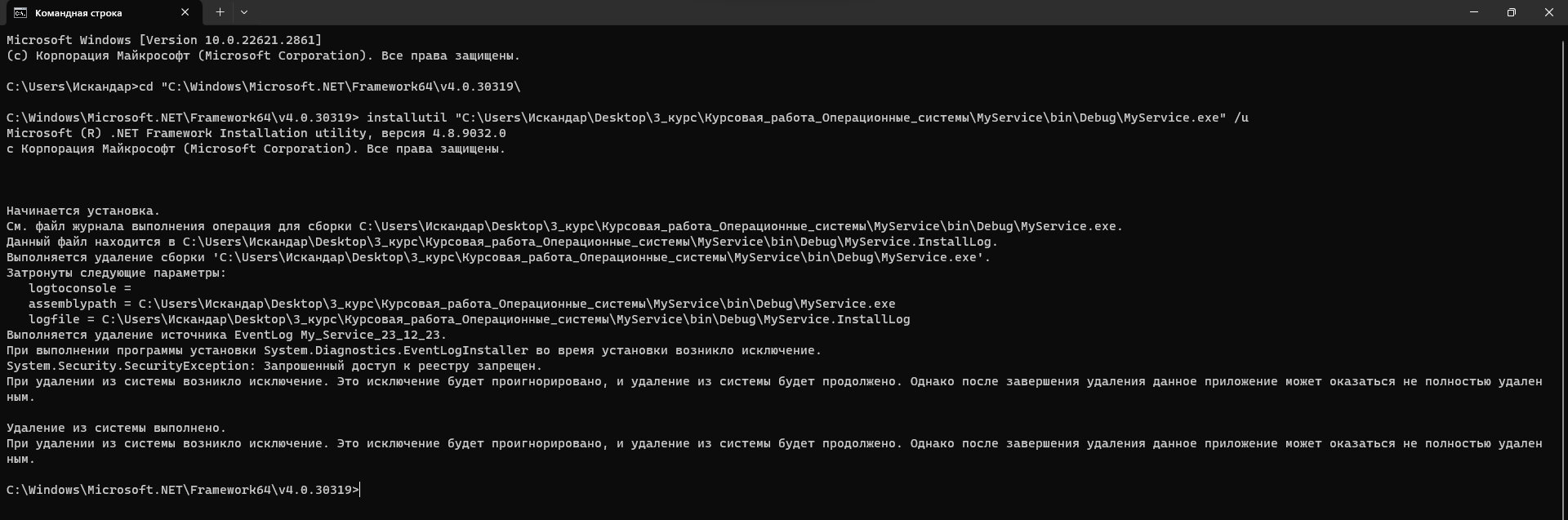


Рисунок 20 – Удаление службы

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана служба, предназначенная для мониторинга изменений в указанном каталоге.   
Созданная служба осуществляет отслеживание создания, удаления и изменения файлов, а также реагирует на событие переименования файла. Реализация службы включает в себя использование FileSystemWatcher для мониторинга, а также запись событий в системный журнал событий и в файл лога.

При желании пользователь может создать собственную службу, функционирующую аналогичным образом, а также данная курсовая работа рассматривает процесс создания служб и предоставляет практический пошаговый пример службы в Windows 11 и Windows 10. Таким образом, она открывает возможность углубленного понимания темы и приобретения практических навыков в области управления службами операционных систем Windows.

**СПИСОК ЛИТЕРАТУРЫ**

1. <https://lifegator.ru/kak-sozdat-sluzhbu-v-windows-10-ili-windows-11/?ysclid=lqjs0gmhn3200357175>
2. <https://remontka.pro/create-service-windows/?ysclid=lqij4ikxiy279384377>
3. <https://www.youtube.com/watch?v=8fsoqYSCfZg&t=989s>
4. <https://learn.microsoft.com/ru-ru/dotnet/framework/windows-services/walkthrough-creating-a-windows-service-application-in-the-component-designer>
5. <https://www.white-windows.ru/kak-sozdat-sluzhbu-v-windows-10-ili-11-sredstvami-sistemy/?ysclid=lqjrz57fuu600324119>