|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_ ***ИУК «Информатика и управление»\_\_\_\_\_\_\_\_\_***

**КАФЕДРА** \_\_ ***ИУК5 «Системы обработки информации»***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

***Планировщик задач***

по дисциплине ***Системное программирование***

Студент гр. ИУК5-41Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Абаев Е.Ю. )

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2021

Калужский филиал   
федерального государственного бюджетного образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»   
(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **\_\_ИУК5\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_(Е.В. Вершинин)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине ***Системное программирование***

Студент\_\_Абаев Е.Ю. ИУК5-41Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, инициалы, индекс группы)

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, инициалы)

График выполнения проекта: 25% к\_4\_нед., 50% к\_7\_нед., 75% к\_10\_нед., 100% к\_14\_нед.

***1. Тема курсового проекта***

***Планировщик задач***

***2. Техническое задание***

*Разработать приложение с использованием функций Windows API для запуска приложений в зависимости от различных критериев (например: в определенное время, при переход системы в определенное состояние, и другие триггеры)* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***3. Оформление курсового проекта***

3.1. Расчетно-пояснительная записка на\_\_\_\_\_\_\_ листах формата А4.

3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О.)

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2021г.

(подпись) (Ф.И.О.)

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**СОДЕРЖАНИЕ**

[Введение.............................................................................................................................................................4](#_Toc72878615)

[1. Техническое задание................................................................................................................................5](#_Toc72878616)

[1.1. Наименование 5](#_Toc72878617)

[1.2. Основание для разработки 5](#_Toc72878618)

[1.3. Исполнитель 5](#_Toc72878619)

[1.4. Цель разработки 5](#_Toc72878620)

[1.5. Содержание работы 6](#_Toc72878621)

[1.5.1. Задачи, подлежащие решению 6](#_Toc72878622)

[1.5.2 Требования к архитектуре АСОИ 6](#_Toc72878623)

[1.5.3. Требования к составу программных компонентов 6](#_Toc72878624)

[1.5.4. Требования к прикладным программам 7](#_Toc72878625)

[1.5.5. Требования к входным/выходным данным 7](#_Toc72878626)

[1.5.6. Требования к временным характеристикам 7](#_Toc72878627)

[1.5.7. Требования к составу технических средств 7](#_Toc72878628)

[1.6. Этапы разработки 8](#_Toc72878629)

[1.7. Техническая документация, предъявляемая по окончании работы 8](#_Toc72878630)

[1.7. Дополнительные условия 9](#_Toc72878631)

[2. Научно-исследовательская часть 10](#_Toc72878632)

[2.1. Постановка задачи проектирования 10](#_Toc72878633)

[2.2. Описание предметной области 10](#_Toc72878634)

[2.3. Анализ аналогов и прототипов 11](#_Toc72878635)

[2.4. Перечень задач подлежащих решению в процессе разработки 14](#_Toc72878636)

[2.5. Обоснование выбора инструментов и платформы для разработки 14](#_Toc72878637)

[3. Проектно-конструкторская часть 17](#_Toc72878638)

[3.1. Разработка структуры приложения 17](#_Toc72878639)

[3.2. Используемые функции Win32API 18](#_Toc72878640)

[3.3. Разработка архитектуры приложения 20](#_Toc72878641)

[3.4. Разработка и реализация алгоритмов приложения 21](#_Toc72878642)

[3.5. Разработка интерфейса взаимодействия пользователя с системой 26](#_Toc72878643)

[4. Проектно-технологическая часть 28](#_Toc72878644)

[4.1. Разработка руководства пользователя 28](#_Toc72878645)

[Заключение......................................................................................................................................................34](#_Toc72878646)

[Список использованной литературы 35](#_Toc72878647)

[Приложение А.................................................................................................................................................36](#_Toc72878648)

[Приложение Б..................................................................................................................................................38](#_Toc72878649)

# Введение

Планировщик заданий — это сервис, который может помочь в автоматизировании работы операционной системы. Среда автоматизации задает параметр включения приложения в определённый момент (к примеру со стартом ОС). Разница между планировщиком заданий и автозагрузкой в этом случае заключается в больших правах для планировщика.

Планировщик заданий служит для настройки автоматических действий при определенных событиях — при включении компьютера или входе в систему, в определенное время, при различных системных событиях и не только.

Как правило, такие задания применяются для автоматизации отдельных процессов:

* параметрическая автоматизация различных задач, выполняемых на компьютере, например:
* автоматическое создание контрольных точек восстановления в определенное время
* очистка диска в определенные дни
* запуск в определенное время дефрагментации диска
* диагностическое тестирование
* оптимизация процесса загрузки компьютера

# Техническое задание

## Наименование

Планировщик задач на основе Windows API.

## Основание для разработки

В процессе эксплуатации системы часто требуется выполнять периодические регламентные задачи обслуживания в автоматическом режиме без вмешательства человека, например, проверку наличия обновлений, создание резервных копий критически важных файлов, выполнения сценариев администрирования и т.п. Планировщик заданий обеспечивает выполнение заранее подготовленных задач в определенное время.

## Исполнитель

Студент группы ИУК5-41Б

## Цель разработки

Целью курсовой работы является формирование практических навыков по разработке и реализации программного приложения с использованием интерфейса прикладного программирования (АРI) операционных систем.

Задачи проектирования:

1. овладение первичными навыками ведения научно-исследовательской, проектной и производственно-технологической деятельности, развитие творческих способностей индивидуально для каждого студента;
2. подготовка к выполнению выпускной квалификационной работы;
3. усвоение методов грамотного ведения, оформления и редактирования технической документации.

## Содержание работы

### 1.5.1. **Задачи, подлежащие решению**

исследование существующих планировщиков задач;

изучение принципов системного программирования и работы WinAPI функций;

реализация методов планирования задач в определенное время;

1. тестирование приложения;
2. подготовка расчетно-пояснительной записки и графических листов;
3. подготовка презентации и речи для защиты курсовой работы;
4. защита курсовой работы.

### 1.5.2 Требования к архитектуре АСОИ

К архитектуре предъявляются следующие требования:

тип приложения – оконное;

модульная архитектура программного обеспечения, в которой каждый модуль представляет собой один из методов планирования задач;

каждый метод должен быть представлен в виде отдельного класса;

### 1.5.3. Требования к составу программных компонентов

Программный комплекс должен состоять из следующих программных компонентов:

1. исполняемый файл PE формата с расширением .exe, реализующий методы планирования задач;
2. полезная нагрузка в виде Shell кода или динамической библиотеки .dll

### 1.5.4. Требования к прикладным программам

Для работы программного комплекса необходимы:

1. Microsoft Windows 8/10;
2. Microsoft Visual Studio 2019;
3. минимальный набор драйверов, обеспечивающих   
   работоспособность ПК.

### 1.5.5. Требования к входным/выходным данным

Входные данные:

1. идентификатор задачи;
2. выбор и настройка времени запуска;
3. путь к нужному файлу;

Выходные данные:

1. запуск выбранного файла в определенное время;
2. всплывающее окно типа «MessageBox» с сообщением об ошибке (если возникла).

### 1.5.6. Требования к временным характеристикам

Требования к временным характеристикам программы   
не предъявляются.

### 1.5.7. Требования к составу технических средств

Для функционирования системы необходимы:

* процессор: 1 ГГц или более быстрый 32-разрядный (x86) или 64-разрядный (x64) процессор;
* RAM: 1 Гбайт (32 бит) или 2 Гбайт (64 бит);
* HDD: 500 Мбайт (32 бит) или 700 Мбайт (64 бит);
* видеокарта: поддержка Microsoft DirectX 9;
* монитор;
* клавиатура;
* мышь.

## **Этапы разработки**

исследование существующих планировщиков задач;

изучение принципов системного программирования и работы WinAPI функций;

выбор методов планирования задач по определенному времени;

реализация методов планирования задач;

тестирование приложения;

## 1.7. **Техническая документация, предъявляемая по окончании работы**

По окончанию работы предъявлена расчетно-пояснительная записка в состав которой входят:

* техническое задание;
* научно-исследовательская часть;
* проектно-конструкторская часть;
* проектно-технологическая часть.

Также должна быть предоставлена графическая часть работы, выполненная формате А1 на 2 листах, в которую входят:

* демонстрационные чертежи;
* алгоритмические схемы.

## **Дополнительные условия**

язык программирования С/С++;

использование Windows API функций для реализации методов планировщика задач;

среда разработки Visual Studio 2019;

тип приложения – оконное;

# Научно-исследовательская часть

## Постановка задачи проектирования

Задача курсовой работы состоит в разработке программы, осуществляющей возможность запланировать запуск программ или скриптов в определённые моменты времени. Необходимо разработать удобный пользовательский интерфейс для работы с программой.

## Описание предметной области

Предметной областью является прикладное программное приложение, осуществляющее запланированный запуск программ в определенное время. Задача которого запускать программы по заданным условиям. Пользователь приложения сможет выбирать время для запуска программы и настраивать его. Приложение должно иметь интерактивный интерфейс для удобного использования пользователем, а конкретнее:

* Иметь подсказки в виде всплывающих окон, чтобы пользователь понимал, как использовать приложение;
* Создание списка запланированных запусков и выбор триггеров для них;
* Просмотр результата работы в виде запущенного приложения в определенное время;

Работа операционной системы Windows - это совокупность сложных процессов. Так, к примеру, весь процесс эксплуатации системы сопровождается периодическим выполнением регламентированных задач, которые обслуживают систему в автоматическом режиме, без ведома и вмешательства пользователя. К таким процессам относится проверка наличия обновлений, создание резервных копий критически необходимых файлов, выполнение процедуры администрирования и многие другие.

Для того, чтобы система спланировала и автоматически выполнила все множество этих операций, существует средство, которое так и называется - планировщик заданий.

Планировщик заданий занимается тем, что заранее подготавливает ряд заданий и сам же запускает их выполнение в определенный промежуток времени. Также планировщик заданий может подстраиваться под событие, которое уже было запущено.

## Анализ аналогов и прототипов

При администрировании компьютеров часто требуется выполнять периодические задачи обслуживания. Планировщик заданий Windows (Рисунок 1) является средством для просмотра и управления запланированными заданиями.

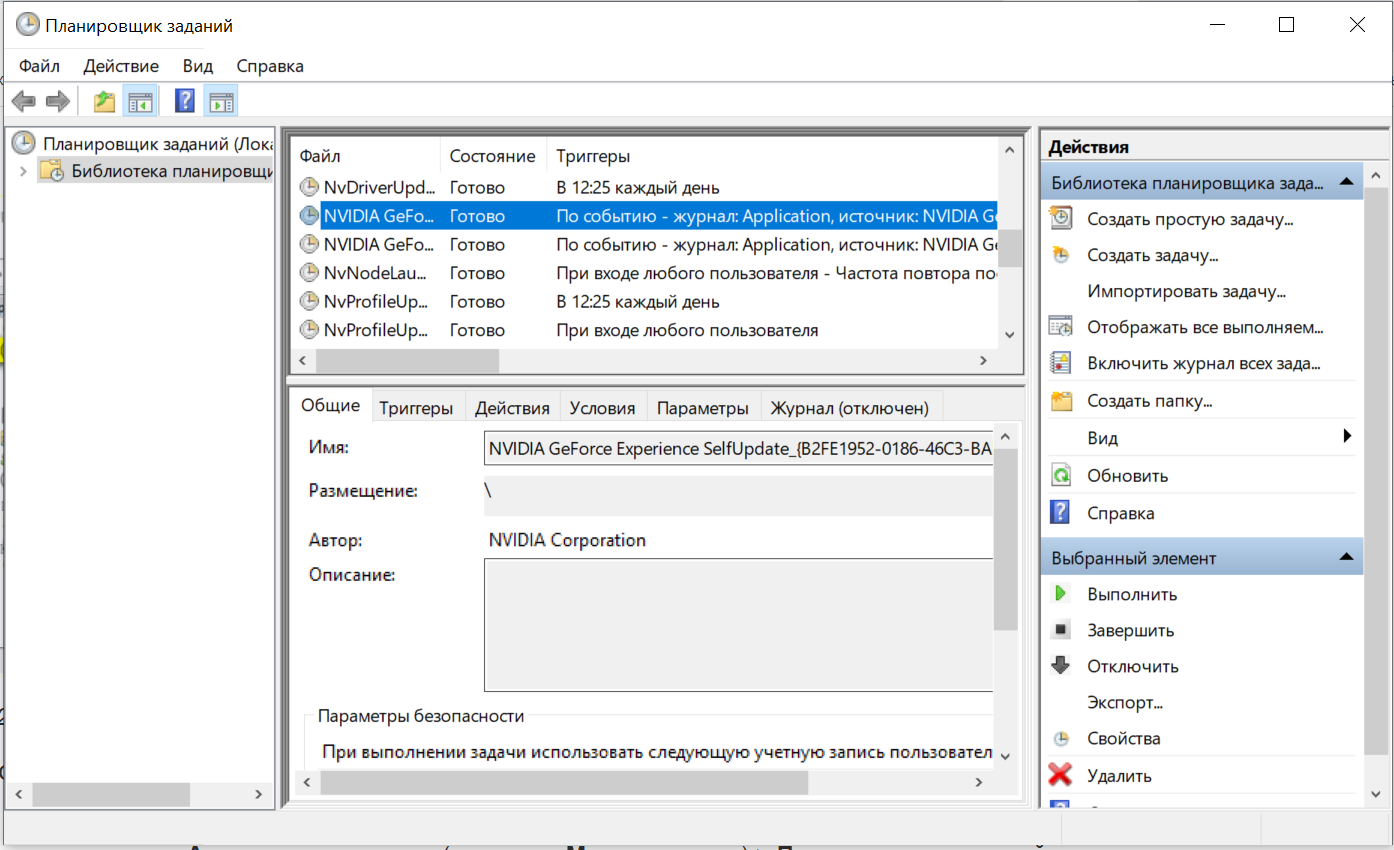


Рисунок 1 – Внешний вид программы “Планировщик заданий”

*Сведения о планировщике задач*

Планировщик задач позволяет автоматически выполнять стандартные задачи на выбранном компьютере. Планировщик задач это достигается за счет мониторинга выбранных критериев (называемых триггерами) и выполнения задач при выполнении этих условий.

Планировщик задач можно использовать для выполнения таких задач, как запуск приложения, отправка сообщения электронной почты или отображение окна сообщения. Задачи могут быть запланированы на выполнение в ответ на эти события или триггеры.

* При возникновении определенного системного события.
* В определенное время.
* В определенное время ежедневное расписание.
* В определенное время по еженедельному расписанию.
* В определенное время по месячному расписанию.
* В определенное время для ежемесячного расписания.
* Когда компьютер переходит в состояние простоя.
* При регистрации задачи.
* При загрузке системы.
* При входе пользователя в систему.
* При изменении состояния сеанса сервера терминалов.

*Задания*

Задача — это запланированная работа, которую выполняет служба планировщик задач. Задача состоит из различных компонентов, но задача должна содержать триггер, который планировщик задач использует для запуска задачи, и действие, описывающее, какая работа планировщик задач будет выполнена.

В следующем списке содержится краткое описание каждого компонента задачи.

* *Триггеры.* Планировщик задач использует триггеры на основе событий или времени, чтобы выяснить, когда следует запускать задачу. Каждая задача может указывать один или несколько триггеров для запуска задачи.
* *Действия.* это действия, фактические работы, выполняемые задачей. Каждая задача может указывать одно или несколько действий для завершения работы.
* *Участники.* Участники определяют контекст безопасности, в котором выполняется задача. Например, участник может определить конкретного пользователя или группу пользователей, которые могут выполнять задачу.
* *Параметры.* Это параметры, которые планировщик задач использует для выполнения задачи в отношении условий, которые являются внешними для самой задачи. Например, эти параметры могут задавать приоритет задачи по отношению к другим задачам, может ли выполняться несколько экземпляров задачи, как задача обрабатывается, когда компьютер находится в состоянии простоя, и другие условия.
* *Сведения о регистрации.* Это административная информация, собираемая при регистрации задачи. Например, эта информация описывает автора задачи, дату регистрации задачи, XML-описание задачи и другие сведения.
* *Данные.* Это дополнительная документация о задаче, предоставленная автором задачи. Например, эти данные могут содержать справку в формате XML, которая может использоваться пользователями при выполнении задачи.

*Триггеры задач*

Триггер — это набор критериев, при достижении которого начинается выполнение задачи. Планировщик задач предоставляет триггеры на основе времени и событий, которые могут запускать задачу несколькими разными способами. Заданную задачу можно запустить с помощью одного или нескольких триггеров. У задачи может быть не более 48 триггеров.

*Триггеры на основе времени*

Триггеры, основанные на времени, запускают задачи в указанное время.

Сюда входит запуск задачи один раз в определенное время или запуск задачи несколько раз для ежедневного, еженедельного, ежемесячного или ежемесячного расписания недели.

*Триггеры на основе событий*

Триггеры на основе событий запускают задачу в ответ на определенные системные события. Например, триггеры на основе событий можно задать для запуска задачи при запуске системы, когда пользователь входит в систему на локальном компьютере или когда система переходит в состояние простоя.[[1]](#Первая_Литра)

## Перечень задач подлежащих решению в процессе разработки

Интерфейс пользователя должен обеспечивать интерактивное взаимодействие при помощи меню, предоставляющего выбор любой из реализованных операций в любой последовательности. В процессе разработки должен быть выделен следующий перечень задач:

* Создание задачи.
* Выбор названия и описания задачи.
* Выбор триггера для данной задачи (конкретную дату и время).
* Выбор пути расположения программы для запуска.
* Выполнение операции должно сопровождаться подсказками в виде всплывающих окон для удобства использования пользователем приложения.

## Обоснование выбора инструментов и платформы для разработки

В операционной системе Windows реализована объектноориентированная идеология. Базовый объект системы – окно, поведение которого определяется методом, называемым функцией окна. Графический образ окна на экране дисплея – прямоугольная рабочая область. Независимо от своего типа любой объект Windows идентифицируется описателем или дескриптором (handle). Все взаимоотношения программного кода с объектом осуществляются только через его дескриптор. Интерфейс прикладного программирования (API – Application Programming Interface) представляет собой совокупность 32-битных функций (Win32 API), которые предназначены для создания приложений (программ), работающих под управлением Microsoft Windows. Функции объявлены в заголовочных файлах. Главный из них − файл windows.h, в котором содержатся ссылки на другие заголовочные файлы.

API — это аббревиатура названия Application Programming Interface (интерфейс прикладного программирования). API представляет собой совокупность функций и инструментов, позволяющих программисту создавать приложения (программы), работающие в некоторой среде.

Win32 API — это набор функций для создания программ, работающих под управлением Microsoft Windows 98, Windows NT или Windows 2000. Все функции этого набора являются 32 битными, что отражено в названии интерфейса.

Для выполнения курсовой работы был выбран язык программирования C++.

**C++**  — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

**C++** широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Одними из отличительных особенностей и очень важных преимуществ C++ является:

* Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщённое программирование, метапрограммирование (шаблоны, макросы).
* Предсказуемое выполнение программ является важным достоинством для построения систем реального времени. Весь код, неявно генерируемый компилятором для реализации языковых возможностей (например, при преобразовании переменной к другому типу), определён в стандарте. Также строго определены места программы, в которых этот код выполняется. Это даёт возможность замерять или рассчитывать время реакции программы на внешнее событие.
* Автоматический вызов деструкторов объектов при их уничтожении, причём в порядке, обратном вызову конструкторов. Это упрощает (достаточно объявить переменную) и делает более надёжным освобождение ресурсов (память, файлы, семафоры и т. п.).
* Пользовательские функции-операторы позволяют кратко и ёмко записывать выражения над пользовательскими типами в естественной алгебраической форме.[[2]](#Вторая_Литра)

# Проектно-конструкторская часть

## Разработка структуры приложения

Для разработки используется среда Microsoft Visual Studio.

Для представления принципа работы ниже приведена UML диаграмма вариантов использования (Рисунок 2).

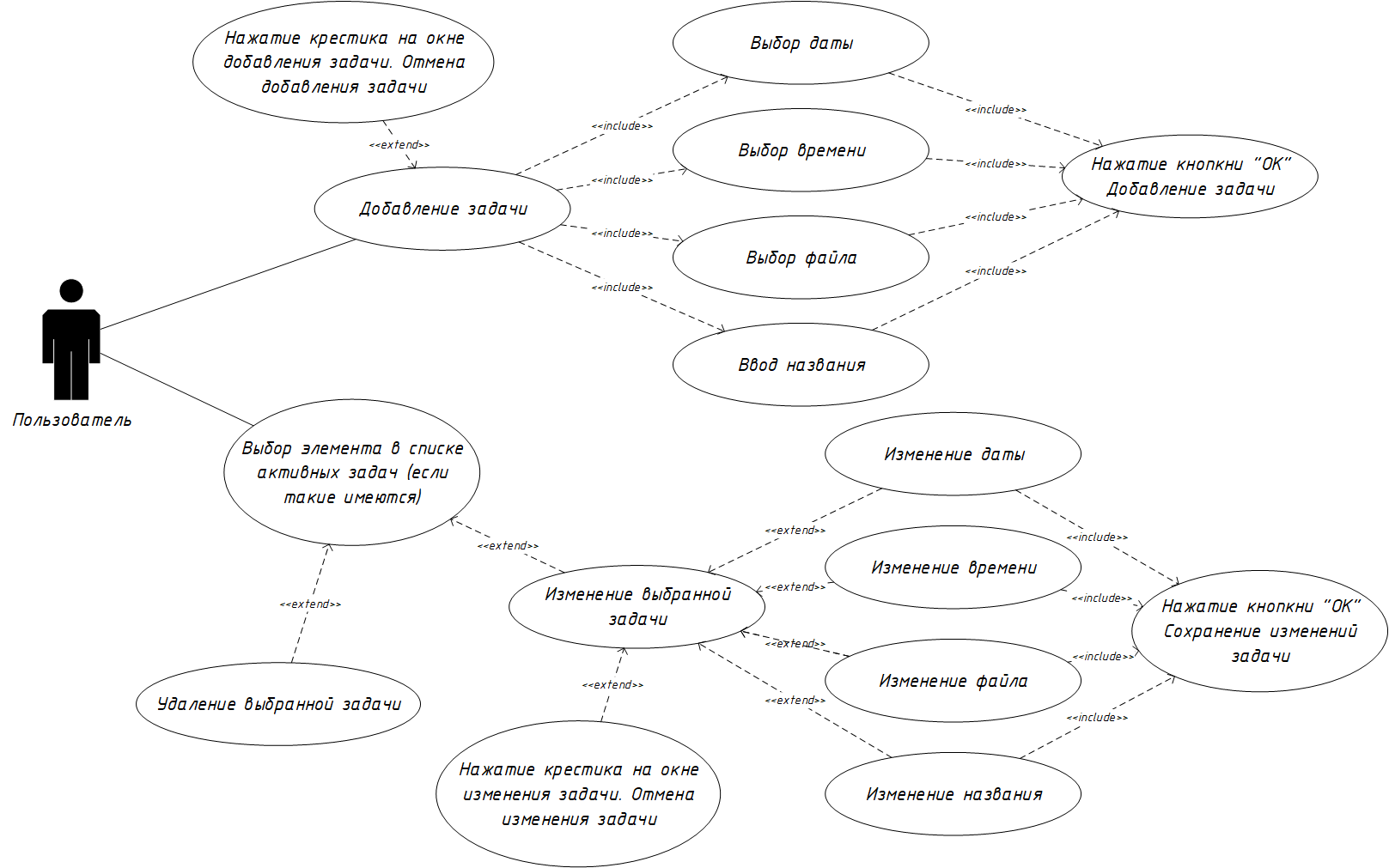


Рисунок 2 - Диаграмма использования программы «Планировщик задач»

Пункт “Добавление задачи” представляет собой кнопку на главном окне, открывающее дочернее окно с полями для заполнения пользователем данных задачи.

Пользователю предоставляется возможность выбрать дату из выпадающего календаря (пункт “Выбор даты”), выбрать время (пункт “Выбор времени”), выбрать файл по нажатию соответствующей кнопки (пункт “Выбор даты”), ввести название для задачи (пункт “Ввод названия”). После этих действий по нажатию кнопки “OK” задача сохраняется. Если нажать на крестик (закрыть окно добавления задачи), то пользователь возвращается в главное окно.

После создания задачи активируется таймер для данной задачи (если задач было несколько, то выбирается ближайшая задача). По истечению таймера открывается нужный файл и вся процедура с выбором ближайшей задачи (если таковы имеются) повторяется.

В главном окне имеется список со всеми активными задачами. Пользователь может выбрать нужную задачу (пункт “Выбор элемента в списке активных задач”) и либо изменить ее (пункт “Изменение выбранной задачи”), либо удалить (пункт “Удаление выбранной задачи”). При изменение открывается все то же окно создания задачи, только уже с заполненными полями, которые пользователь по желанию может изменить. По нажатию кнопки “ОК” все изменения сохраняются и задача активируется.

## Используемые функции Win32API

Используемые библиотеки Win32API, функции и их аргументы:

1. **User32.lib**

* **SetTimer**(

хендл главного окна,

ID таймера,

задержка для таймера,

NULL) – данная функция используется для установки таймера запуска предложения. NULL в последнем аргументе означает, что таймер будет обрабатываться в сообщении WM\_TIMER.

* **CreateWindow**(

класс окна,

текст окна,

стили окна, x, y, ширина, высота, хендл родительского окна, ID окна, nullptr, nullptr) - данная функция используется для создания контроллов в окнах. От класса окна зависит какой контролл будет создан (Button, Edit, Static, ListBox) этим стилям соответствуют стили окна.

* **CreateWindowEx**() – в данной функции аргументы те же что у и CreateWindow(). Данная функция используется для создания DATETIMEPICK\_CLASS – контроллы выбора даты и времени (зависит от параметров в стиле окна)
* **DestroyWindow**(хендл окна) – данная функция используется для закрытия окна создания и изменения задачи
* **EnableWindow**(хендл окна, BOOL) – данная функция используется для включения или отключения мыши и ввода с клавиатуры (зависит от параметра BOOL) в определенном окне (Окно указывается в параметре “Хендл окна”)
* **ShowWindow**(хендл окна, SW\_SHOWDEFAULT) – данная функция используется для показа окна
* **UpdateWindow**(хендл окна) – данная функция используется для перерисовки окна
* **SendMessage**(хендл адресата,

Сообщение,

Дополнительная информация,

Дополнительная информация) – данная функция одна из самых основных. Она используется, например, для установки шрифтов в окнах, для получения данных, введенных пользователем, для установки значений в ListBox, для закрытия дочерних окон.

* **GetDlgItem**(хендл окна с контроллом, ID контролла) – данная функция используется для получения данных из контролла (например получение названия задачи, которое ввел пользователь)
* **SetDlgItemText**(хендл главного онка, ID контролла, текст) **-** даннаяфункция используется для установки текста контролла. В данном приложении используется для изменения текста label контроллов.

1. **Shell32.lib**

* **ShellExecute**(

NULL,

L"open",

путь файла,

NULL, NULL, SW\_SHOWNORMAL) – Данная фукнция используется для открытия файла по выбранному пользователем пути. Эта функция активируется по срабатыванию таймера в WM\_TIMER.

1. **Kernel32.lib**

* **GetLocalTime**(указатель на структуру SYSTEMTIME) – данная функция используется для получения локального времени.

## Разработка архитектуры приложения

Данное приложение использует монолитный архитектурный шаблон. Данный шаблон часто используется, т.к. он очень прост и не принуждает разделять приложение на отдельные независимые части.

## Разработка и реализация алгоритмов приложения

Общий алгоритм работы приложения (Рисунок 4).



Рисунок 4 – Алгоритм работы приложения

Основные алгоритмы приложения:

1. bool valid\_NOW\_date - проверка даты на актуальность (Рисунок 5).



Рисунок 5 – Проверка даты

1. bool valid\_NOW\_time – аналогичная функция для проверки времени на актуальность (Рисунок 6).



Рисунок 6 – Проверка времени

1. bool valid\_NOW\_DATETIME – проверка и времени и даты (объединение предыдущих двух функций) (Рисунок 7).



Рисунок 7 – Проверка времени и даты

1. bool check\_DATE\_TIME\_ADRESS – Полная проверка введенных пользователем данных (Рисунок 8).



Рисунок 8 – Проверка времени, даты, пути к файлу и названия

1. void LauchTask – запуск таймера задачи (Рисунок 9).



Рисунок 9 – Запуск таймера задачи

## Разработка интерфейса взаимодействия пользователя с системой

Главное окно программы [(Приложение А)](#приложение1) должно состоять из кнопки для создания задачи, лист-бокса для отображения задачи, лейбл-текста для отображения информации о выбранной задачи, кнопки удалить и изменить выбранную задачу (Рисунок 10).



Рисунок 10 – Макет главного окна приложения

При нажатии на кнопку создания и изменения задачи (при выбранной активной задачи в лист-боксе) создается дочернее окно [(Приложение Б)](#приложение2) (Рисунок 11) с полями для ввода названия задачи, выпадающем календарем с выбором даты, поле с выбором времени, и кнопка для выбора файла (которая открывает классическое окно выбора файла) .



Рисунок 11 – Макет окна создания и изменения задачи

Функции, используемые в создании элементов интерфейса в окнах (подробно данные функции описаны в пункте 3.2):

* CreateWindow() – для кнопок, лист-бокса, лейблов иё поля ввода названия
* CreateWindowEx() – для полей выбора даты и времени

# Проектно-технологическая часть

## Разработка руководства пользователя

Запуск программы осуществляется по открытию файла TaskScheduler.exe

После запуска открывается главное окно приложения (Рисунок 12.)

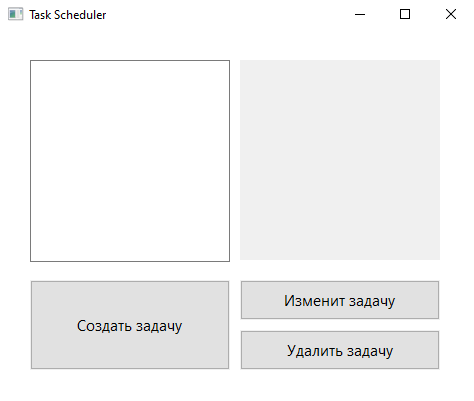


Рисунок 12 – Главное окно приложения

В верхней части окна расположен список активных задач, и слева от него – информация о выбранной задачи. В нижней части окна находятся три кнопки:

* “Создать задачу” – создание задачу.
* “Изменить задачу” – изменение выбранного элемента в списке активных задач.
* “Удалить задачу” – удаление выбранного элемента в списке активных задач.

Для создания задачи нужно нажать на кнопку создания задачи “Create Task”. После нажатия открывается окно создания задачи (Рисунок 13).

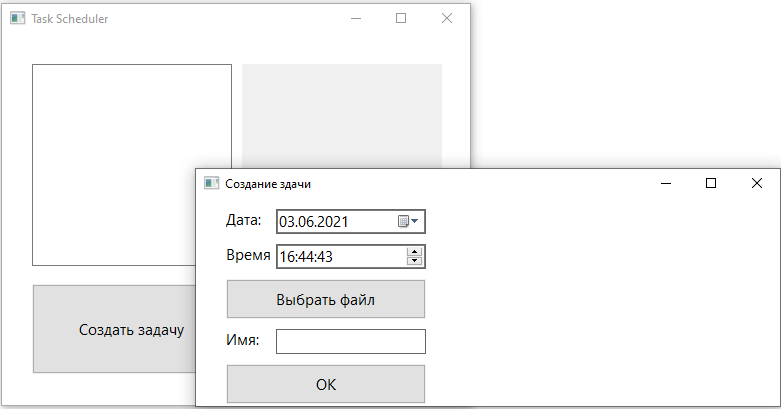


Рисунок 13 – Окно создания задачи

В поле “Date” выбирается дата с помощью календаря (Рисунок 14). Дата должна быть актуальной.

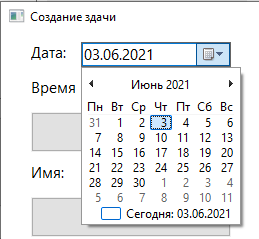


Рисунок 14 – Календарь для выбора даты

В поле “Time” выбирается время с помощью стрелочек с левой стороны поля или с помощью ввода с клавиатуры.

При выборе неактуальной комбинации времени и даты (меньше, чем системное) и нажатии кнопки “OK”, показывается сообщение об ошибке (Рисунок 15).

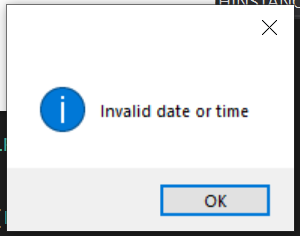


Рисунок 15 – Ошибка при неправильном выборе времени и даты

По нажатию кнопки “Choose File” открывается окно выбора файла (Рисунок 16). После выбора нужно подтвердить выбор по нажатию кнопки “Открыть”. Далее можно изменить выбор, нажав еще раз кнопку “Choose File”.

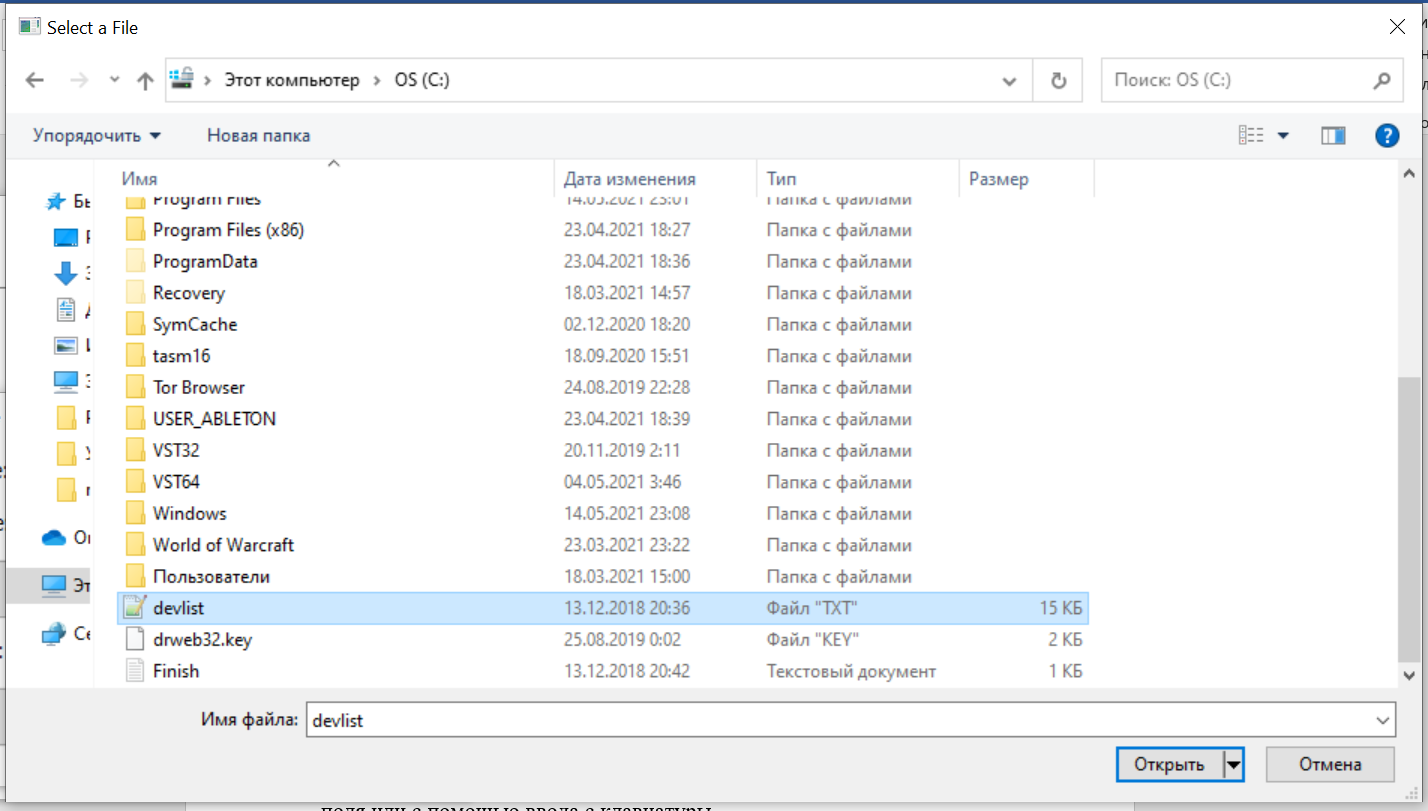


Рисунок 16 – Окно выбора файла

Если файл не был выбран и нажата кнопка “OK”, то высвечивается следующее сообщение (Рисунок 17).

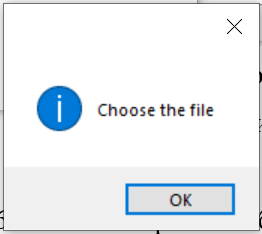


Рисунок 17 – Ошибка при не выбранном файле

В поле “Title” пользователем вводится название задачи, которое будет отображаться в списке активных задач на главном окне.

Если название не было введено и нажата кнопка “OK”, то высвечивается следующее сообщение (Рисунок 18).

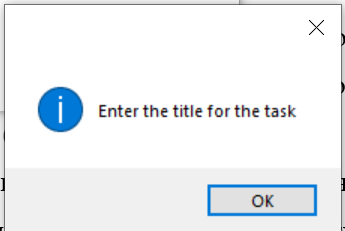


Рисунок 18 – Ошибка при не введенном названии

После заполнения всех полей нужно нажать кнопку “OK”, после чего задача появится на главном окне. При закрытии окна до нажатии кнопки “OK” задача не сохранится.

Данную задачу можно выбрать нажатием левой кнопки мыши по названию в списке и просмотреть информацию о ней (Рисунок 19).

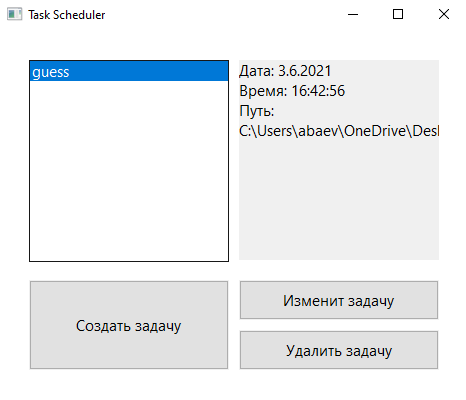


Рисунок 19 – Выбранная активная задача на главном окне

Выбранную задачу можно изменить, нажав кнопку “Change selected task”. При нажатии на данную кнопку откроется аналогичное окно создания задачи, только с заполненными полями (Рисунок 20).

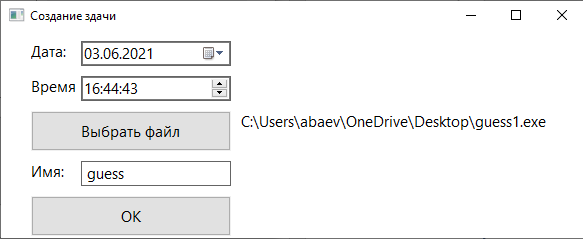


Рисунок 20 – Изменение выбранной задачи

В данном окне можно изменить любое поле и по нажатию кнопки “OK” – все изменения сохраняются (Рисунок 21). При закрытии окна до нажатии кнопки “OK” изменения не сохраняются.

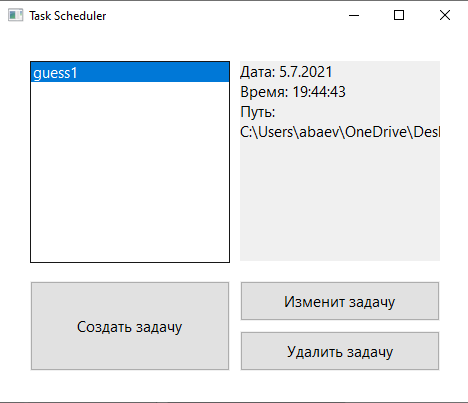


Рисунок 21 – Просмотр измененной задачи

При некорректном времени, при не выбранном файле и при не введенном названии появляются те же сообщения с ошибками, как и при создании задачи.

Выбранную задачу можно удалить по нажатию кнопки “Delete selected task”.

С наступлением времени, выбранного пользователем, задача выполняется и открывает соответствующий этой задаче файл. После выполнения задачи она исчезает из списка активных задач. После выполнения всех задач появляется сообщение “All tasks completed”.

В данном приложении создан понятный, упрощенный и интуитивный интерфейс для максимального комфорта работы пользователя с ней.

Главное окно и дополнительные окна выполнены в стандартных цветах для прикладного приложения Windows (белый, серый и синий).

Элементы интерфейса сгруппированы по значимости. Наиболее важная кнопка “Create Task” на главном окне увеличена.

# Заключение

В ходе выполнения данной курсовой работы был разработан планировщик задач на основе WinAPI.

Работа выполнялась в несколько этапов: была разработана структура системы и реализовано прикладное оконное приложение на основе WinAPI.

Были сформированы навыки по разработке и реализации программного приложения с использованием интерфейса прикладного программирования (АРI) операционных систем.

В данной работе выполнены все поставленные задачи.

Спроектированное приложение подойдет для задач, в ходе которых необходимо запустить файл в заданное пользователем время.

В будущем можно усовершенствовать разработанное приложение путем добавления дополнительного функционала (например добавление дополнительных условий для создаваемых задач).

# Список использованной литературы

1. <https://docs.microsoft.com/ru-ru/windows/win32/taskschd/task-scheduler-start-page>

2. <http://window.edu.ru/catalog/pdf2txt/389/67389/40509>

# Приложение А

Создание главного окна:

int CALLBACK wWinMain(HINSTANCE hInstance, HINSTANCE, PWSTR szCmdLine, int nCmdShow)

{

static MSG msg{};

static HWND hwnd{}, hChildWnd{}, hChildWnd\_EDIT{}, hEditMAIN1{}, hListboxMAIN{}, hLabelmain{}, hChildDatePick{}, hChildTimePick{}, hChildEdit1{}, hChildEdit2{}, hChildEditName{}, hChildButton2{}, hChildButton1{}, hChildLabel1{}, hChildLabel\_EDIT1{}, hChildLabel\_EDIT2{}, hChildLabelName{};

WNDCLASSEX wc{ sizeof(WNDCLASSEX) };

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hbrBackground = reinterpret\_cast<HBRUSH>(GetStockObject(WHITE\_BRUSH));

wc.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wc.hIcon = LoadIcon(nullptr, IDI\_APPLICATION);

wc.hIconSm = LoadIcon(nullptr, IDI\_APPLICATION);

wc.hInstance = hInstance;

wc.lpfnWndProc = [](HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam) -> LRESULT

{

switch (uMsg)

{

//Создание контролов на родительском окне

case WM\_CREATE:

{ ... }

return 0;

//Срабатывание таймера

case WM\_TIMER:

{ ... }

return 0;

//Обработка нажатия контроллов главного окна

case WM\_COMMAND:

{ ... }

return 0;

//Закрытие родительского окна

case WM\_DESTROY:

{ ... }

return 0;

}

return DefWindowProc(hWnd, uMsg, wParam, lParam); // вызывается в случае если сообщение не обрабатывается

};

wc.lpszClassName = L"MyAppClass";

wc.lpszMenuName = nullptr;

wc.style = CS\_VREDRAW | CS\_HREDRAW;

if (!RegisterClassEx(&wc))

return EXIT\_FAILURE;

if (hwnd = CreateWindow(wc.lpszClassName, L"Task Scheduler", WS\_OVERLAPPEDWINDOW, 0, 0, 490, 410, nullptr, nullptr, wc.hInstance, nullptr); hwnd == INVALID\_HANDLE\_VALUE)

return EXIT\_FAILURE;

ShowWindow(hwnd, nCmdShow); // показ окна

UpdateWindow(hwnd); // перерисовка окна

while (GetMessage(&msg, nullptr, 0, 0)) // Цикл обработки сообщений

{

TranslateMessage(&msg); // функция расшифровывает системное сообщение

DispatchMessage(&msg); // функция передаёт сообщение в оконную процедуру на обработку

}

return static\_cast<int> (msg.wParam); // возвращаемое значение точки входа

}

# Приложение Б

Создание дочерних окон

//AddWindow - Создание дочернего окна

static pair<bool, HWND> AddWindow(const wstring&& winClass, const wstring&& title, HWND hParentWnd, const WNDPROC callback)

{

UnregisterClass(winClass.c\_str(), GetModuleHandle(nullptr));

WNDCLASSEX wc{ sizeof(WNDCLASSEX) }; // Эта структура отвечает за некие х-ки окна (в фигурных скобках размеры).Исп. агрегатная инициализация.

HWND hWindow{};

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hbrBackground = reinterpret\_cast<HBRUSH>(GetStockObject(WHITE\_BRUSH));

wc.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wc.hIcon = LoadIcon(nullptr, IDI\_APPLICATION);

wc.hIconSm = LoadIcon(nullptr, IDI\_APPLICATION);

wc.lpfnWndProc = callback;

wc.lpszClassName = winClass.c\_str();

wc.style = CS\_VREDRAW | CS\_HREDRAW;

const auto create\_window = [&hWindow, &winClass, &title, &hParentWnd]() -> pair<bool, HWND> {

if (hWindow = CreateWindow(winClass.c\_str(), title.c\_str(), WS\_OVERLAPPEDWINDOW, 100, 100, 600, 270, hParentWnd, nullptr, nullptr, nullptr); !hWindow)

return { false,nullptr };

ShowWindow(hWindow, SW\_SHOWDEFAULT); // показ окна

UpdateWindow(hWindow);

return { true,hWindow };

};

if (!RegisterClassEx(&wc))

return create\_window();

return create\_window();

}