|  |  |
| --- | --- |
| **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:**

***Разработка прикладного программного приложения для управления автозапуском в ОС Windows на основе Windows API***

по дисциплине ***Системное программирование***

Студент гр.\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Макаренко С.С.)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Фролов П.В.)

(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)

«8» февраля 2021г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине **Системное программирование**

Студент Макаренко С.С. ИУК5-41Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_\_\_\_

(фамилия, инициалы, индекс группы)

Руководитель Фролов П.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(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***

***Разработка приложения для управления автозапуском ОС Windows***

***2. Техническое задание***

Разработать прикладное программное приложение для управления автозапуском в ОС Windows на основе Windows API

***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 «8» февраля 2021г.

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_/\_\_\_\_ Фролов П.В. \_

(подпись) (Ф.И.О.)

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_ \_/\_Макаренко С.С.\_\_/ «8» февраля\_2021 г. \_

(подпись) (Ф.И.О.)

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc71836176)

[Техническое задание 4](#_Toc71836177)

[Научно-исследовательская часть 9](#_Toc71836178)

[Проектно-конструкторская часть 20](#_Toc71836179)

[Проектно-технологическая часть 37](#_Toc71836180)

[Источники разработки 49](#_Toc71836181)

Техническое задание

1. Наименование

Прикладное программное приложение для управления автозапуском в ОС Windows на основе Windows API.

2. Основание для разработки

Автозапуск в Windows является широко используемой функцией. В автозагрузку добавляются программы, которые необходимы пользователю постоянно при работе компьютера, например, антивирус. В Windows 10 автозагрузка добавлена в диспетчер задач. В настоящее время существует множество программ для управления автозапуском в ОС Windows, однако в большинстве из них отсутствует функция добавления любой программы в автозагрузку. Очистка автозагрузки зачастую применяется именно на низкопроизводительных компьютерах, следственно и программа для редактирования автозагрузки должна использовать как можно меньше ресурсов компьютера.

Существуют такие способы автозапуска как системная автозагрузка, планировщик задач, реестр и т.д. Как правило, во всех существующих аналогах пользователю не предоставляется выбора, каким образом добавить приложение в автозапуск. Для того, чтобы предоставить гибкий инструмент для автозапуска различных приложений любым из доступных способов, необходимо разработать прикладное программное обеспечение, которое предоставляет пользователю выбор способа добавления программы в автозагрузку OC Windows с последующей возможностью удаления.

3. Исполнитель

Студент группы ИУК5-41Б Макаренко С.С.

4. 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.

Целью разработки является программное обеспечение для управления автозапуском в OC Windows.

5. Содержание работы

5.1. Задачи, подлежащие решению:

1. исследование предметной области;
2. реализация пользовательского интерфейса;
3. реализация актуальных методов управления автозапуском;
4. тестирование реализованных методов;
5. исправление допущенных ошибок;
6. подготовка расчетно-пояснительной записки и графических листов;
7. подготовка презентации и речи для защиты курсовой работы;
8. защита курсовой работы.

5.2. Требования к архитектуре АСОИ

К архитектуре предъявляются следующие требования:

1. модульная архитектура программного обеспечения, в которой каждый модуль представляет собой один из методов управления автозагрузкой;
2. каждый метод должен быть представлен в виде отдельного класса;
3. приложение должно представлять собой многопоточное приложение;
4. работа пользовательского интерфейса должна быть реализована в основном потоке, а все методы автозапуска в отдельных потоках.

5.3. Требования к составу программных компонентов

Программный комплекс должен состоять из следующих программных компонентов:

1. исполняемый файл PE формата с расширением .exe, реализующий управление автозагрузкой, а именно добавление и удаление программ

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.

5.5. Требования к входным/выходным данным

Входные данные:

1. путь к исполняемому файлу или окно выбора программы, которую необходимо добавить в автозагрузку;
2. метод автозагрузки.

Выходные данные:

1. список приложений, добавленных в автозагрузку с указанием примененного метода.

5.6. Требования к временным характеристикам

Требования к временным характеристикам программы не предъявляются.

5.7. Требования к составу технических средств

Для функционирования системы необходимы:

* процессор: 1 ГГц и быстрее;
* 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 с драйвером WDDM;
* монитор;
* клавиатура;
* мышь.

6. Этапы разработки

1. реализация пользовательского интерфейса;
2. реализация актуальных методов управления автозагрузкой в ОС Windows;
3. тестирование методов
4. исправление выявленных ошибок
5. финальное редактирование пользовательского интерфейса

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 А3 на 2 листах, в которую входят:

* демонстрационные чертежи;
* алгоритмические схемы.

8. Дополнительные условия

1. язык программирования С/С++;
2. использование Windows API функций для реализации методов внедрения программного кода в сторонний процесс;
3. тип приложения – оконное;
4. интерфейс должен предоставлять пользователю выбор для добавления программ в автозапуск или удаления программ из автозапуска;
5. добавление программ должно производиться как по пути расположения программы, так и с помощью графического интерфейса;
6. у пользователя должна быть возможность выбора метода автозагрузки, также этот метод должен отображаться в списке программ, находящихся в автозагрузке
7. при выборе удаления должен появляться список всех программ, находящихся в автозагрузке с кнопкой отключения автозагрузки для каждой программы.

Научно-исследовательская часть

1. Постановка задачи проектирования.

Задачей проектирования данной курсовой работы является разработка прикладного программного приложения для управления автозапуском в ОС Windows на основе Windows API, предназначенного для добавления программ в автозагрузку, с последующей возможностью их удаления. Так же необходимо разработать удобный пользовательский интерфейс для работы с программой.

1. Описание предметной области.

Программы в автозагрузке — это то программное обеспечение (ПО), которое запускается при входе в систему и может служить для самых разных целей: это антивирус, мессенджеры, сервисы облачного хранения данных — для многих из них существуют значки в области уведомлений справа внизу. Однако, точно так же в автозагрузку могут добавляться и вредоносные программы. Более того, даже избыток «полезных» элементов, запускаемых автоматически, может приводить к тому, что компьютер работает медленнее, в этом случае стоит удалить из автозагрузки какие-то необязательные из них.

Одинаково осуществляется завершение работы приложений. Перед завершением работы необходимо сохранить все данные и закончить работу.

При анализе выбранной предметной области было проанализировано несколько способов решения поставленной задачи.

Следует отметить, что существует несколько способов добавления программ в автозапуск, а именно: добавление с помощью планировщика заданий Windows, добавление ярлыка программы в папку автозагрузки Windows, добавление в реестр пользователя и добавление в общий реестр операционной системы Windows.

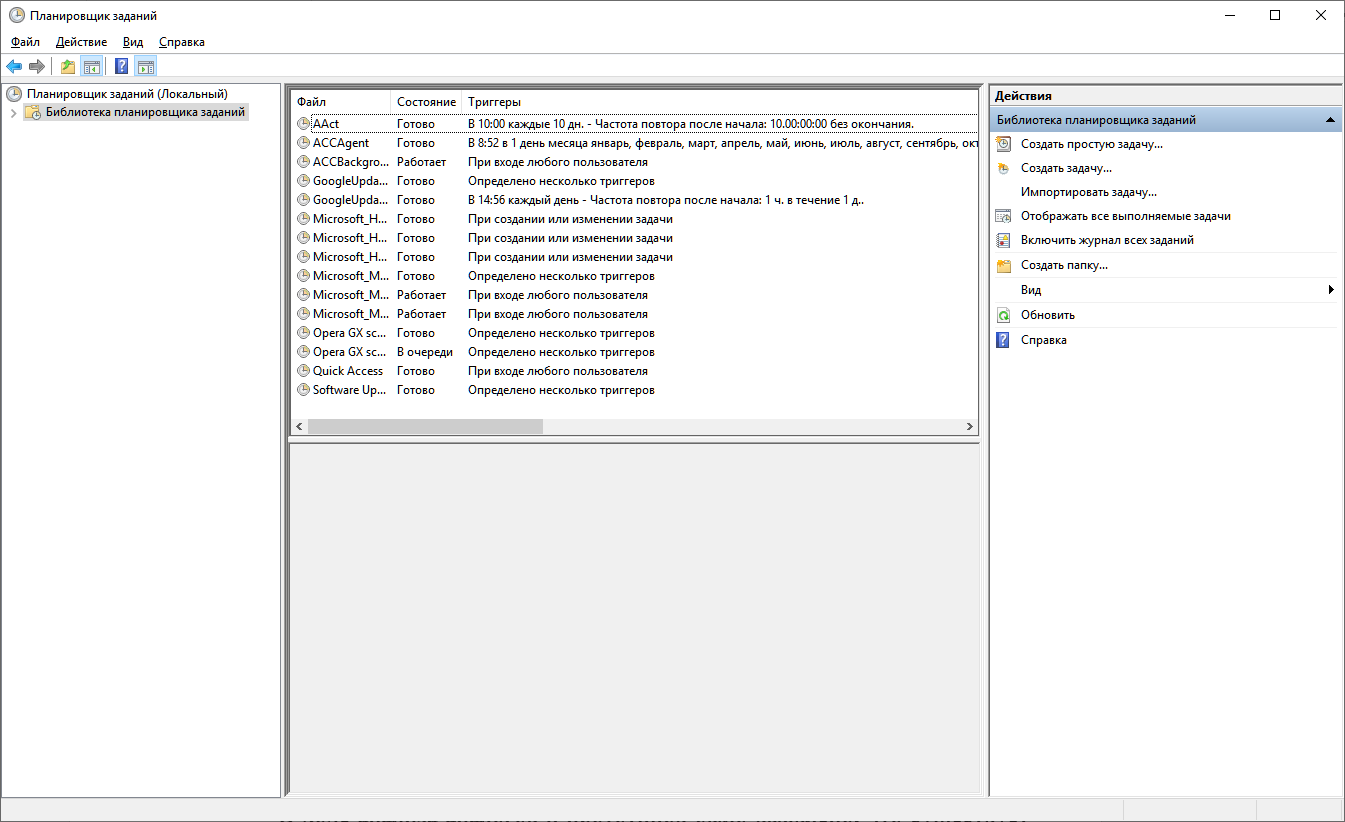


Рисунок 1. Добавление с помощью планировщика задач Windows.

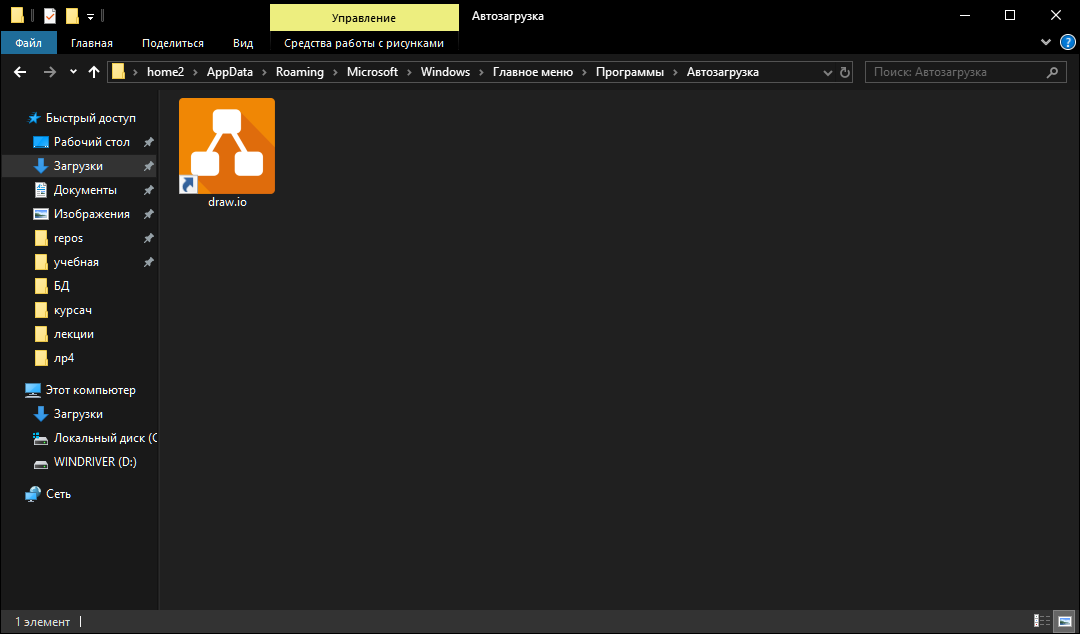


Рисунок 2. Добавление ярлыков в папку автозагрузки.

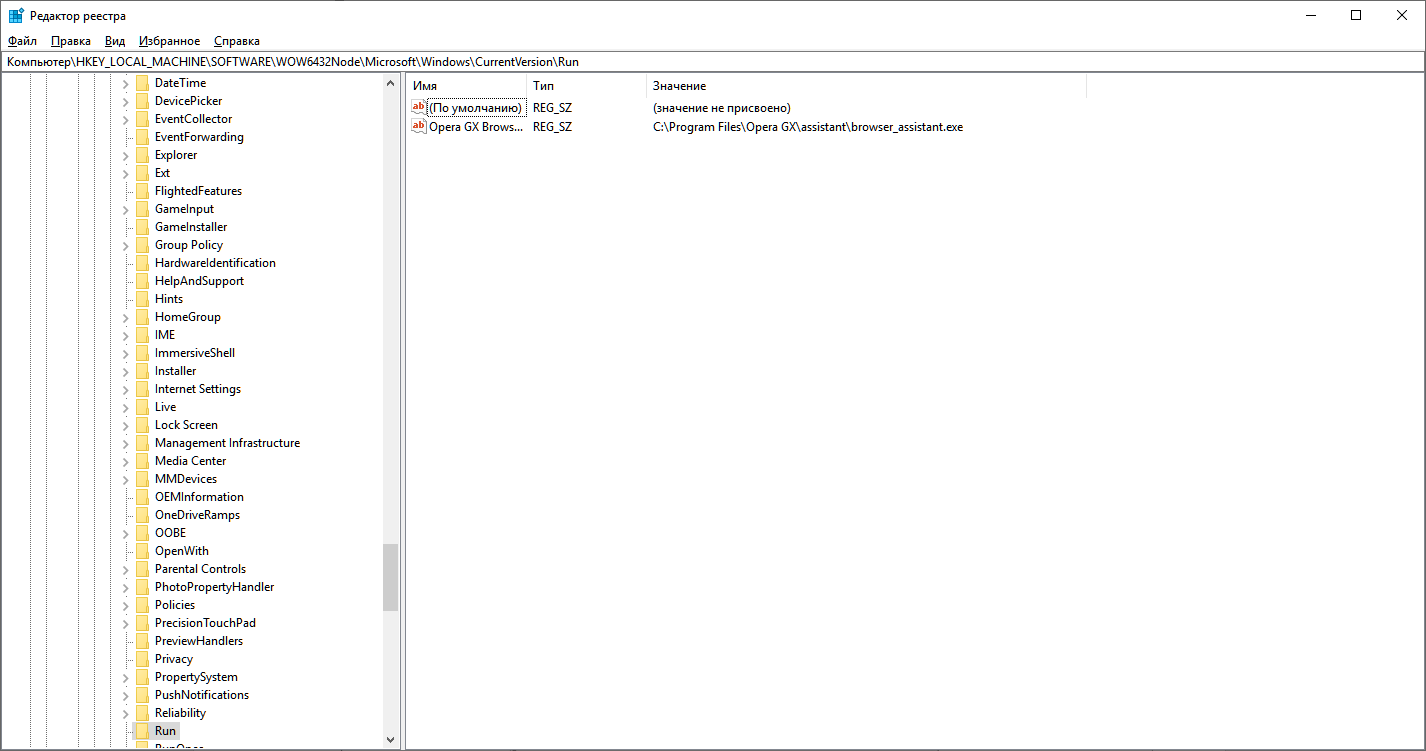


Рисунок 3. Добавление в общий реестр.

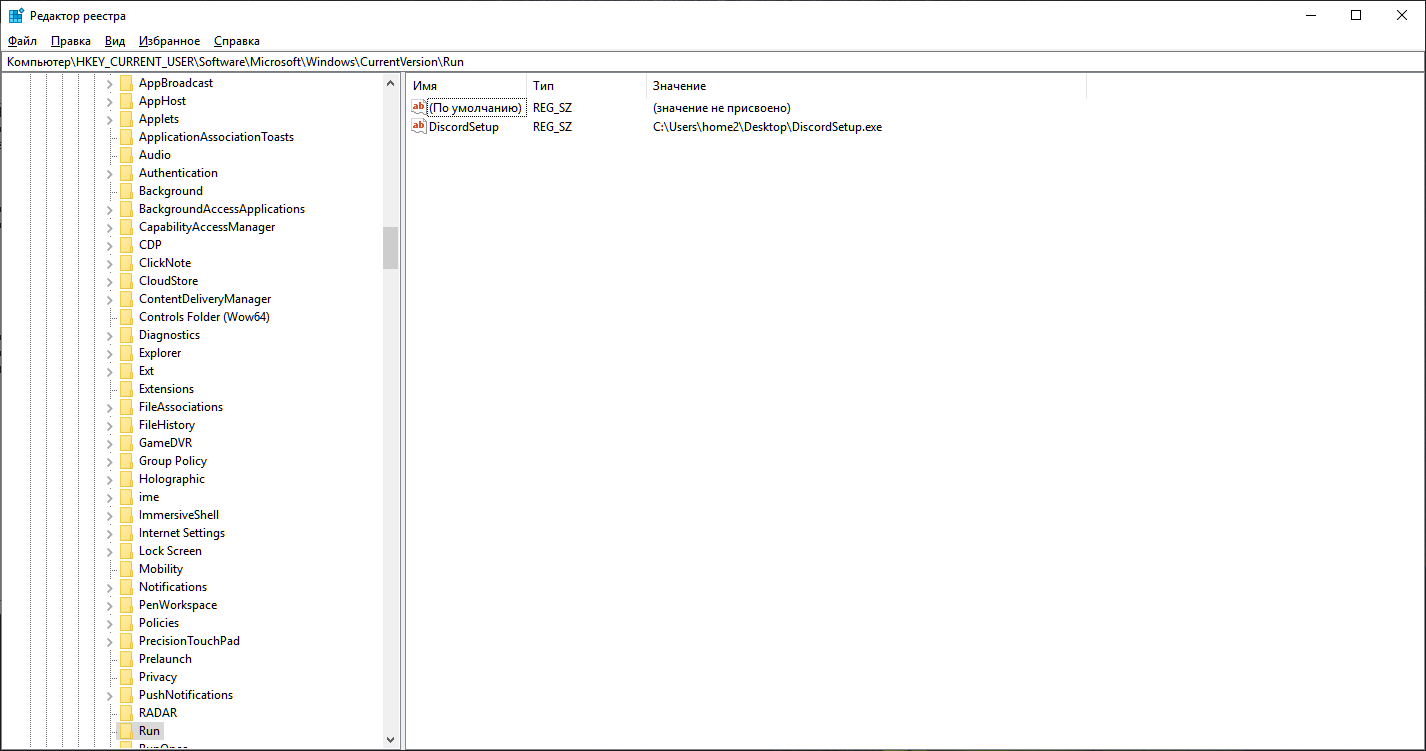


Рисунок 4. Добавление в реестр учётной записи.

Но все эти методы управления подходят не всем пользователям. Да и совершенно неудобно искать программу, автозапуск которой необходимо отключить во всех этих методах.

1. Анализ аналогов и прототипов.

Поэтому существует множество программ для управления автозагрузкой, вот некоторые из них:

1. В Windows 10 возможность отключения программ из автозагрузки в диспетчере задач. Но в диспетчере задач отображаются далеко не все программы, находящиеся в автозагрузке. Также отсутствует возможность добавления программ в автозагрузку.

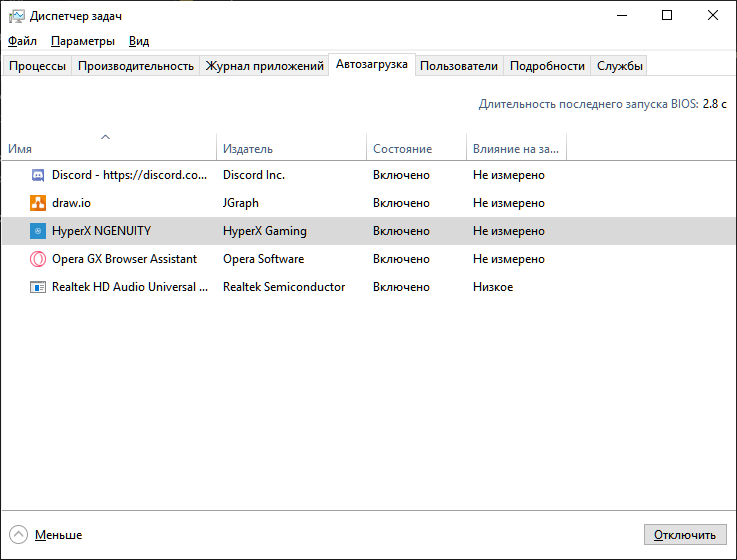


Рисунок 5. Автозагрузка в диспетчере задач.

1. Win 10 Tweaker. В этой программе отображается большинство программ, находящихся в автозагрузке, но всё равно не все. И снова отсутствует возможность добавления программ в автозапуск.



Рисунок 6. Автозагрузка в программе Win 10 Tweaker.

1. Некоторые производители ноутбуков также предоставляют свои программы в которых присутствует возможность управления автозагрузкой. Например, программа Acer Care Center: в данной программе также отображены не все автозагрузки и отсутствует возможность добавления программ в автозагрузку.

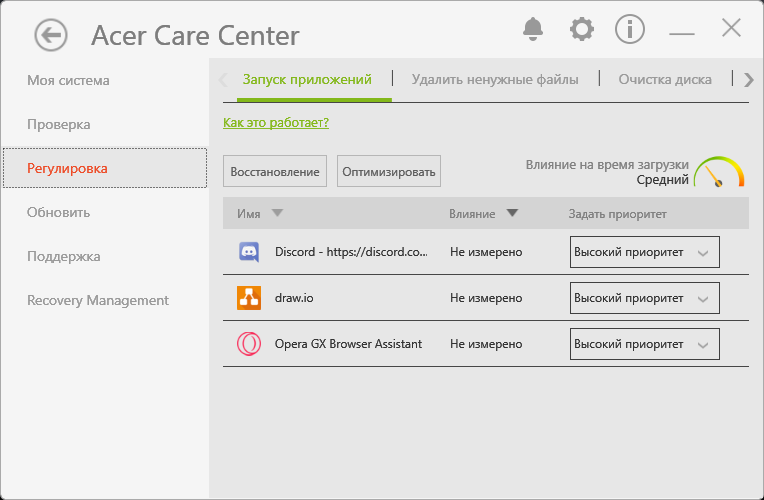


Рисунок 7. Автозагрузка в программе Acer Care Center.

1. Перечень задач, подлежащих решению в процессе разработки.

Исходя из результатов анализа существующих аналогов было принято решение разработать собственное ПО реализующее работу с автозагрузкой. Интерфейс пользователя должен представлять пользователю возможности: просмотра всех программ, находящихся в автозапуске; добавления программ; удаления программ из автозапуска. В процессе разработки должен быть выделен следующий перечень задач:

* получение списка всех автозагрузок из папки автозагрузки, планировщика задач и всех разделов реестра;
* удаление программ из автозапуска;
* возможность отключения и включения автозапуска для программ, добавленных в автозагрузку с помощью планировщика;
* удобный интерфейс для добавления программы в автозапуск;
* добавление программы в автозагрузку всеми доступными способами;
* Реализовать всплывающие окна с предупреждениями при неверных действиях пользователя;
* При реализации необходимо предусмотреть проверку на корректность введенных данных;

Также нужно протестировать и отладить все этапы процесса разработки ПО.

1. Обоснование выбора инструментов и платформы для разработки.

Microsoft Windows (англ. windows — окна) — семейство операционных систем корпорации Майкрософт (Microsoft), ориентированных на применение графического интерфейса при управлении. Изначально были представлены многофункциональными надстройками для MS-DOS.

В настоящее время Windows является одной из самых распространенных операционных систем в мире. По официальному сообщению, компании Microsoft, к середине 2020 года ОС Windows 10 теперь установлена на один миллиард персональных компьютеров, согласно информации компании о количестве активных устройств (ноутбуков, ПК и планшетов) с этой ОС.

По состоянию на апрель 2020 года, операционная система Windows заняла второе место в общей мировой статистике использования ОС. Ее распространение составило 32,31% от количества всех операционных систем, существующих в настоящее время. Первое место занимает ОС для смартфонов Android.

По статистике ОС для компьютеров операционная система Windows заняла первое место. Ее распространение составило 76,58% от количества всех ОС. Тогда как OS X — 18,93 и Linux — 1,62%.

Общий рейтинг операционных систем в России, включая десктопы, мобильные, планшеты и игровые приставки, показывает, что лидером также является Windows, который установлен на 55,58% устройств. Операционные системы, установленные на компьютерах в России также в большей мере на Windows — 83,83%.

Стиль программирования Windows-приложений принципиально отличается от того, который сложился в операционных системах раннего поколения.

В MS-DOS программа монопольно владеет всеми ресурсами системы и является инициатором взаимодействия с операционной системой.

Совсем иначе дело обстоит в операционной системе Windows, которая строилась как многозадачная, и именно операционная система является инициатором обращения к программе. Все ресурсы Windows являются разделяемыми, и программа (приложение), не может владеть ими монопольно. В связи с такой идеологией построения операционной системы приложение должно ждать посылки сообщения операционной системы и лишь после его получения выполнить определенные действия, затем вновь перейти в режим ожидания очередного сообщения

Windows генерирует множество различных сообщений, которые направляются приложению, например, щелчок кнопки мыши или нажатие клавиши на клавиатуре. Если приложение не обрабатывает какие-то сообщения, реакция на них осуществляется операционной системой стандартным способом, так что задачей программиста является обработка лишь тех сообщений, которые необходимы приложению.

Разработчиками операционной системы Windows была создана библиотека функций, при помощи которых и происходит взаимодействие приложения с операционной системой, так называемые функции Программного интерфейса приложений (Application Program Interface, API).

Библиотека API-функций разрабатывалась в расчете на то, что ее можно использовать для любого языка программирования.

Вся идеология построения Windows-приложения ориентирована на взаимодействие с пользователем.

Благодаря удобным механизмам взаимодействия с операционной системой и распространённостью ОС Windows, была выбрана эта операционная система.

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

Язык С++ является одним из самых распространенных языков.

C++ — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование.

С++ — это язык программирования общего назначения, хорошо известный своей эффективностью, экономичностью, и переносимостью. Указанные преимущества С++ обеспечивают хорошее качество разработки почти любого вида программного продукта. Использование С++ в качестве инструментального языка позволяет получать быстрые и компактные программы.

С++ сочетает эффективность и мощность в относительно малом по размеру языке. Хотя С++ не содержит встроенных компонент языка, выполняющих ввод-вывод, распределение памяти, манипуляций с экраном или управление процессами, тем не менее, системное окружение С++ располагает библиотекой объектных модулей, в которой реализованы подобные функции. Библиотека поддерживает многие из функций, которые требуются.

Это решение позволяет изолировать языковые особенности от специфики процессора, на котором выполняется результирующая программа. Строгое определение языка делает его независимым от любых деталей операционной системы.

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.

* WinAPI спроектирован для написания прикладных программ, предназначенных для работы под управлением операционной системы MS Windows.
* Работа через Windows API — это наиболее близкий к операционной системе способ взаимодействия с ней из прикладных программ. Более низкий уровень доступа, необходимый только для драйверов устройств, в текущих версиях Windows предоставляется через Windows Driver Model.
* Windows API представляет собой множество функций, структур данных и числовых констант, следующих соглашениям языка Си. В то же время конвенция вызова функций отличается от cdecl, принятой для языка C: Windows API использует stdcall (winapi).

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

В итоге разработка приложения будет производиться в среде Visual Studio на C++ с использованием WinAPI под ОС Windows.

Проектно-конструкторская часть

1. Разработка структуры системы.

Приложение состоит из трёх основных частей: добавление приложений в автозапуск, просмотр и удаление этих приложений.

Для просмотра и удаления приложений из автозапуска нужно собрать данные из всех методов автозапуска, а именно: папка «Автозагрузка», реестр, «Планировщик заданий».

Для удаления программ из автозапуска нужно учитывать метод автозапуска выбранной программы и в соответствии с методом производить удаление или отключение автозапуска. Нужно реализовать удаление ярлыков или файлов из папки «Автозагрузка», удаление строк из реестра, удаление или отключение заданий из планировщика заданий.

Для добавления программ в автозапуск необходимо предоставить пользователю выбор метода добавления. реализовать создание ярлыков в папке автозапуска. Нужно реализовать добавление ярлыков выбранной программы в папку автозапуска, добавление строк в реестр, создание заданий в планировщике заданий.

Все компоненты разрабатываются только на Windows 10 в такой среде разработки, как Visual Studio 2019, используя С/С++ в качестве языка программирования.

1. Разработка алгоритмов обработки информации.

В программе должны быть функции считывания программ, находящихся в автозагрузке:

* StartupRead – возвращает список файлов из папки Startup
* TasksRead – возвращает список задач планировщика
* RegRead – возвращает список по имени и пути из реестра

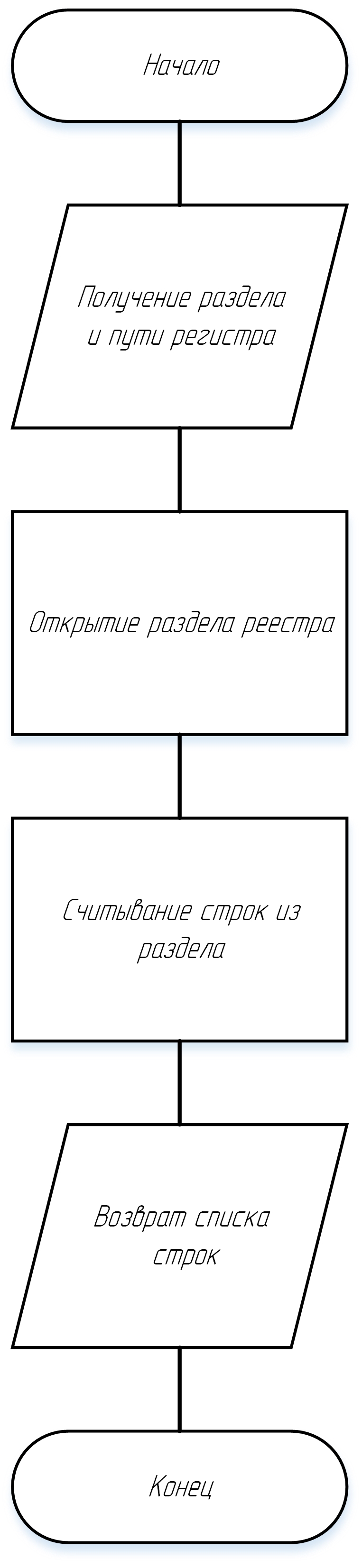


Рисунок 8. Алгоритм работы функции StartupRead.

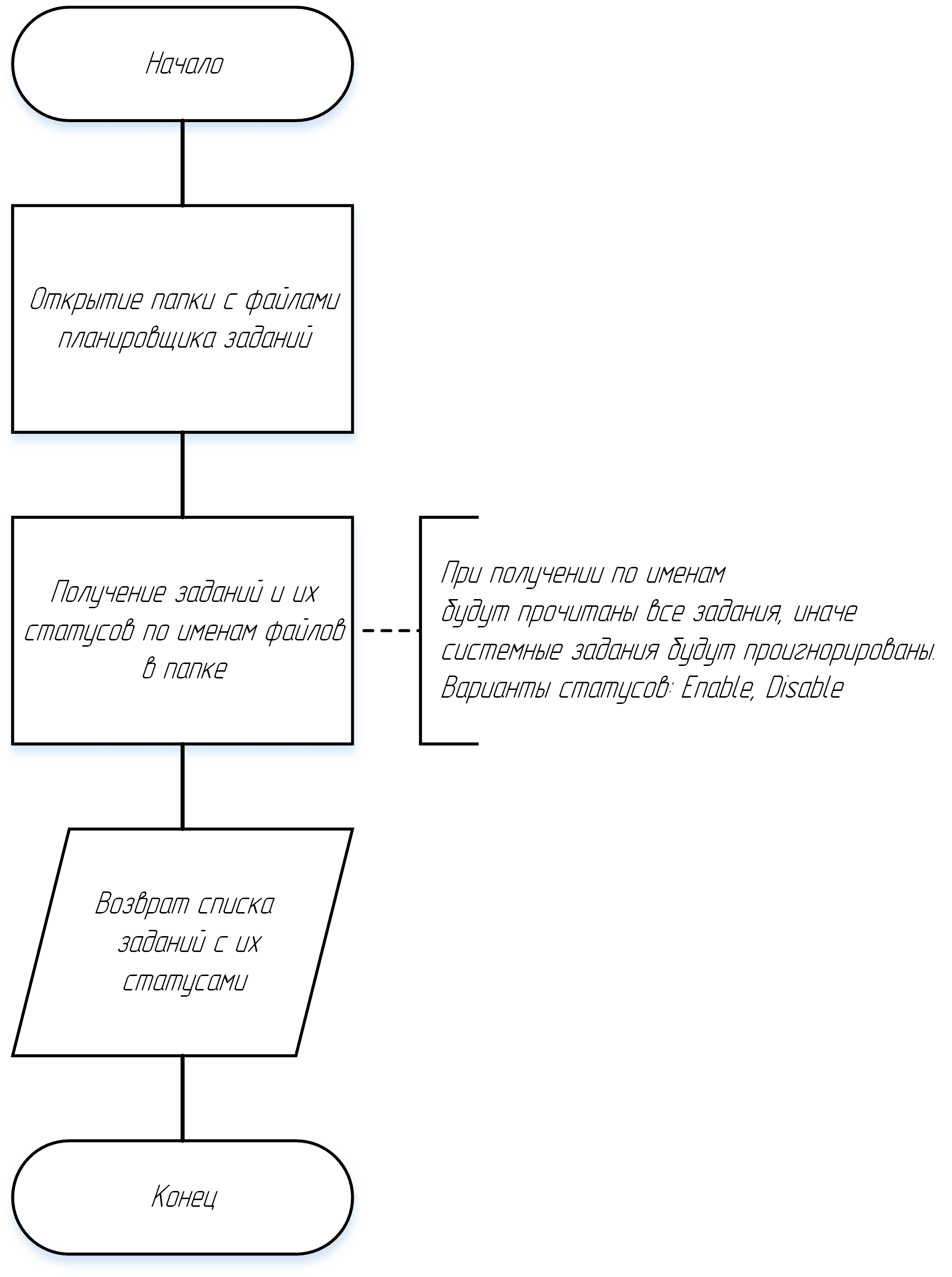


Рисунок 9. Алгоритм работы функции TasksRead.

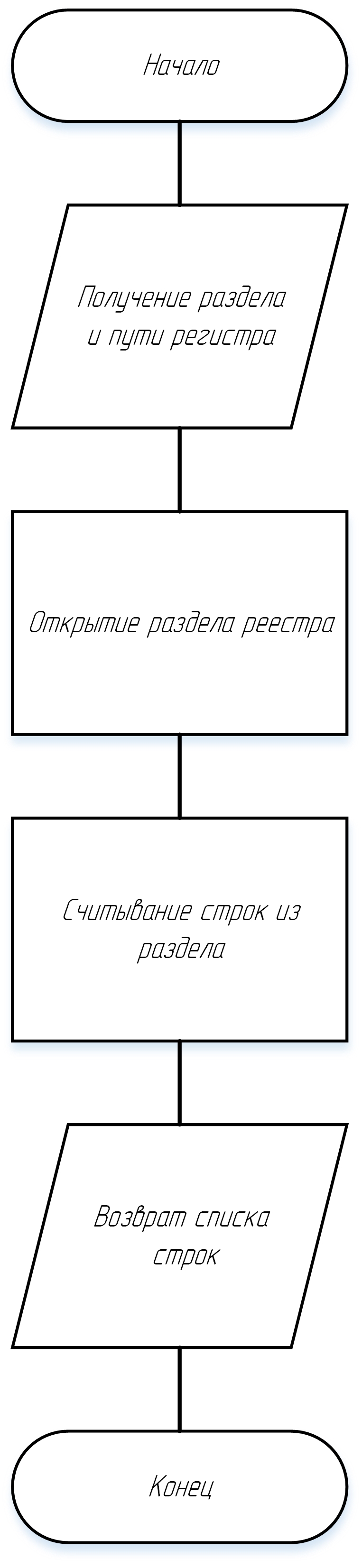


Рисунок 10. Алгоритм работы функции RegRead.

Каждая функция чтения возвращает строку со списком данных программ находящихся в автозагрузке в форме: «название программы \t метод автозагрузки \t enable/disable \n»

Третий столбец заполнятся только для автозагрузки методом «Планировщик заданий»

Также необходимы функции удаления программ из автозапуска:

* StartupDel – удаляет программу из папки автозагрузки
* RegDel – удаляет программу из реестра автозагрузки
* DeleteTask – удаляет задачу из планировщика
* On\_Off\_Task – включает и выключает задачу в планировщике



Рисунок 11. Алгоритм работы функции StartupDel.



Рисунок 12. Алгоритм работы функции RegDel.

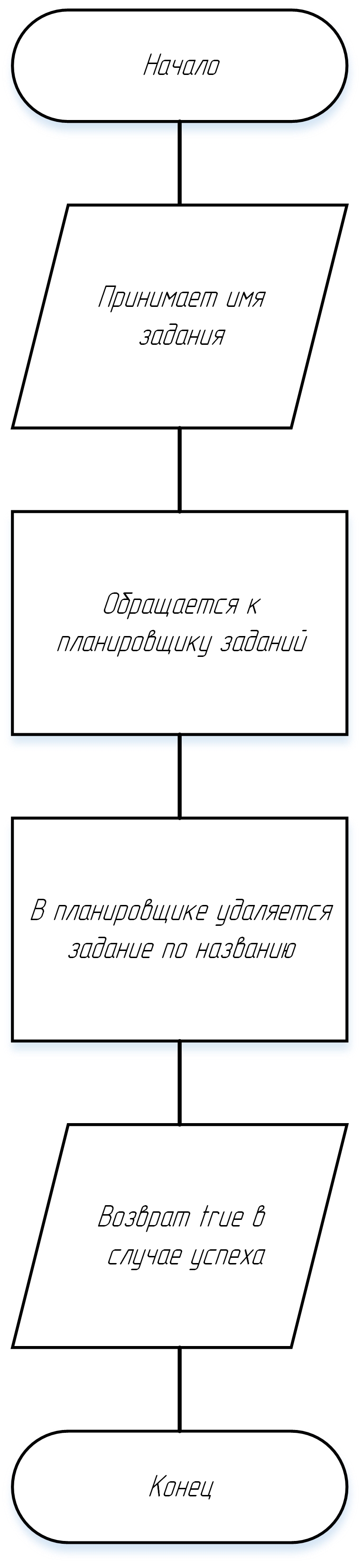


Рисунок 13. Алгоритм работы функции DeleteTask.

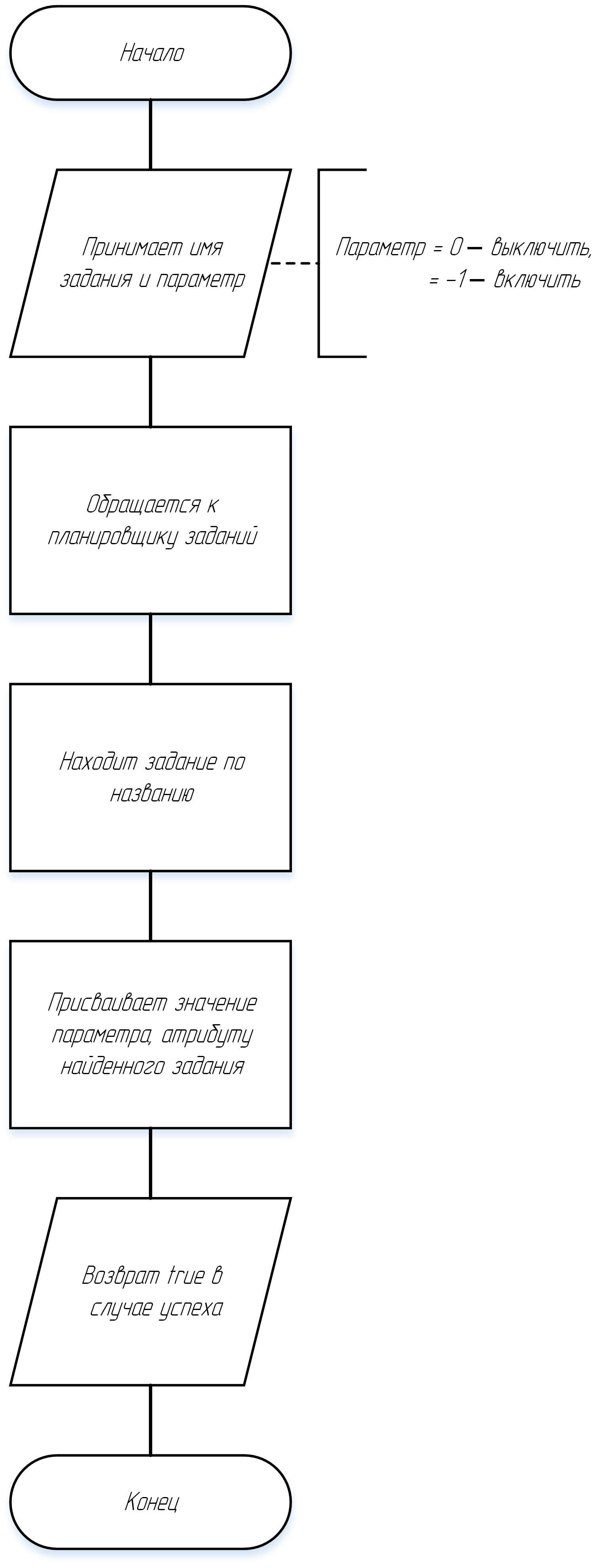


Рисунок 14. Алгоритм работы функции On\_Off\_Task.

И функции добавления:

* CreateLink – создаёт ярлык
* RegAdd – добавляет строки в реестр
* CreateTask – создаёт задачу в планировщике заданий



Рисунок 15. Алгоритм работы функции CreateLink.

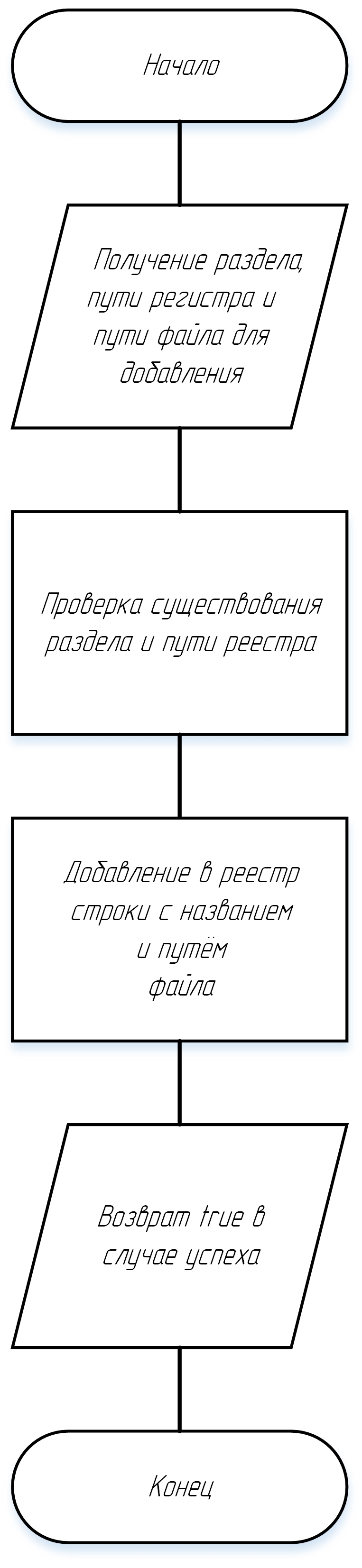


Рисунок 16. Алгоритм работы функции RegAdd.

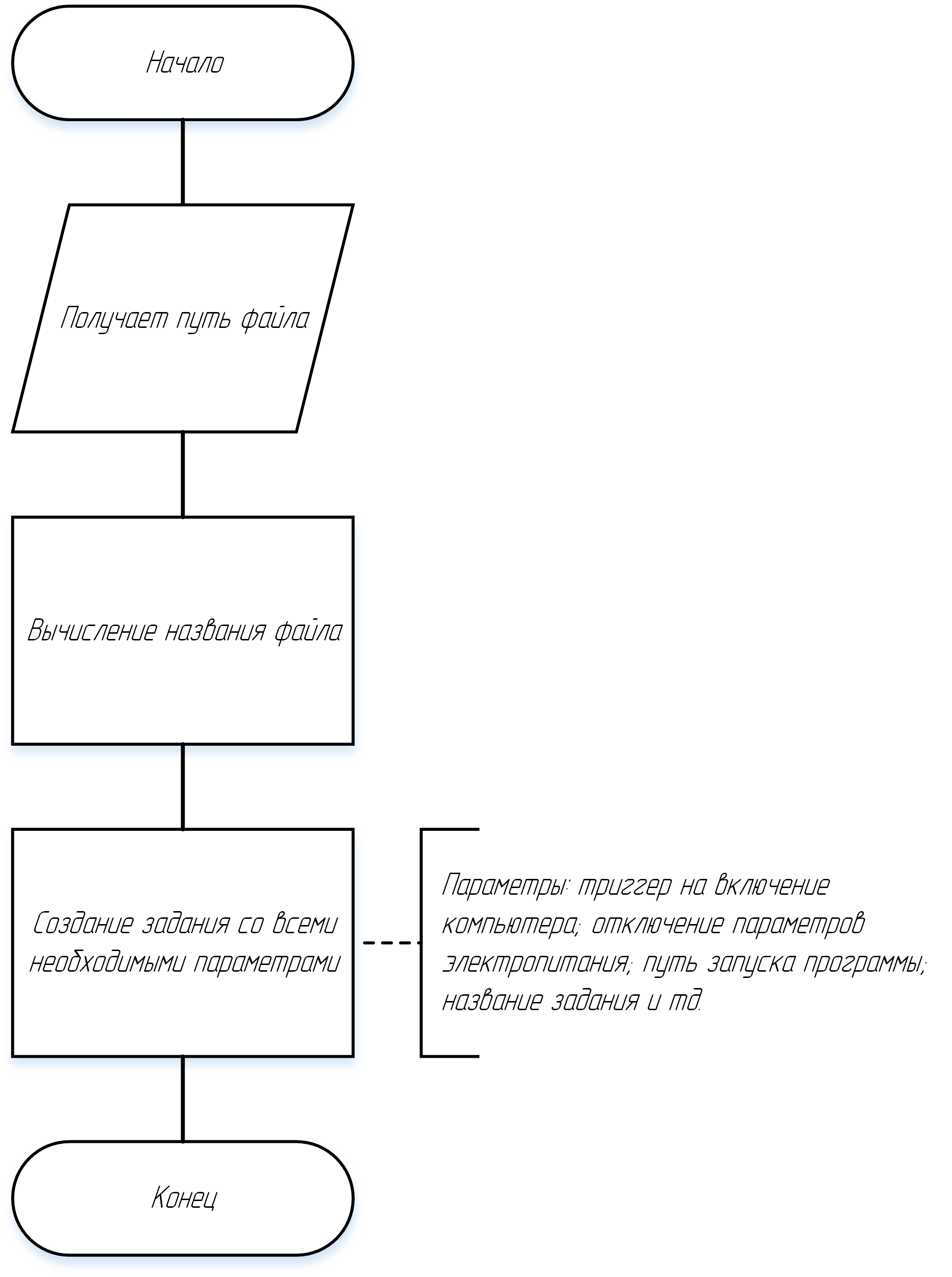


Рисунок 17. Алгоритм работы функции CreateTask.

В главном окне приложения должен быть обработчик сообщений в котором должны обрабатываться сообщения:

* Выход из программы case WM\_DESTROY;
* Перерисовка интерфейса при изменении размера окна case WM\_PAINT;
* Нажатия клавиатуры case WM\_KEYDOWN;
* Нажатия кнопок case ID\_Btn1 и тд;

1. Разработка архитектуры приложения.

Для сохранения структуры приложения для каждого метода автозагрузки должен быть отдельный класс с необходимыми функциями. Все методы и функции классов должны быть статическими, для использования их без создания экземпляров классов.

Все методы и функции должны запускаться в отдельных потоках, чтобы исключить подвисание интерфейса.

В самом начале при запуске приложения ещё до начала отрисовки и всех остальных процессов в отдельном потоке вызывается метод GetAllAutoStarts() заполняющий переменную AllAutoStarts. После завершения работы этого метода происходит перерисовка окна приложения.

Все функции добавления и удаления программ тип bool, которые возвращают true при успехе, иначе false. Это необходимо для вывода сообщения пользователю об успехе или ошибке

При сборе данных о всех программах, находящихся в автозапуске каждая функция должна запускаться в отдельном потоке, для улучшения быстродействия программы. Все возвращенные строки собираются в одну и

При выборе программы добавленной, методом планировщика будет появляться диалоговое окно с выбором: отключить, включить, удалить.

1. Реализация готового приложения.

Внутри проекта один исходный файл CourseWork.cpp. Он содержит весь необходимый код для приложения, а именно:

* константы стилей;
* вспомогательные переменные и переменные окон;
* методы void GetAllAutoStarts() для записи всех переменных в переменную AllAutoStarts в отдельном потоке,

и void CreateListBox() для создания и заполнения ListBox’а в отдельном потоке;

* функция WinMain в которой происходит регистрация основного окна, его создание и отображение, а также цикл обработки, который отправляет сообщения в WndProc();
* функция регистрации класса главного окна RegMyWindowClass;
* и функции обработки сообщений окон: WndProc, Wind2, Wind3.

И заголовочные файлы: framework.h, Resource.h, Tools.h, Startup.h, Registry.h, Tasks.h.

В файле framework.h содержаться все используемые библиотеки, пространства имён и макроопределения, для удобности их подключения к другим файлам проекта.

В файле Resource.h содержатся макроопределения кодов всех элементов интерфейса.

В файле Tools.h содержится класс Tools с публичными, статическими вспомогательными функциями, такими как:

static wstring StrToWstr(string str) – переводит string в wstring

static string WstrToStr(wstring wstr) – переводит wstring в string

static wstring GetStartup() – возвращает расположение папки «Автозагрузка»

static wstring GetFileName(LPCWSTR path) – возвращает имя файла по пути

static wstring SelectFile(HWND hWnd) – диалог выбора файла, возвращает путь

В файле Startup.h содержится класс Startup c функциями необходимыми для работы с автозапуском методом папки «Автозагрузка»

static wstring StartupRead() – возвращает список файлов из папки Startup

static bool StartupDel(wstring wstr) – удаляет программу из папки автозагрузки

static bool CreateLink(LPCWSTR lpszPathObj, wstring path) – создаёт ярлык

В файле Registry.h содержится класс Registry с функциями, которые нужны для работы с реестром

static bool RegAdd(HKEY hkeyPath, LPCWSTR lpszPathObj, LPCWSTR path) – добавляет строки в реестр

static wstring RegRead(HKEY key, LPCWSTR str) – возвращает список по имени и пути из реестра

static bool RegDel(HKEY key, LPCWSTR str, wstring name) – удаляет программу из реестра автозагрузки

В файле Tasks.h содержится класс Tasks с функциями работы с планировщиком заданий Windows

static bool CreateTask(LPCWSTR lpszPathObj) – создаёт задачу в планировщике заданий

static bool DeleteTask(LPCWSTR lpszPathObj) – удаляет задачу из планировщика

static wstring TasksRead() – возвращает список задач планировщика

static bool On\_Off\_Task(LPCWSTR lpszPathObj, VARIANT\_BOOL varbool) – включает и выключает задачу в планировщике

Также при разработке данного приложения были использованы WinAPI функции, такие как:

* CreateWindow – создает перекрытое, всплывающее или дочернее окно.
* CreateThread – создает поток для выполнения в виртуальном адресном пространстве вызывающего процесса.
* DestroyWindow – уничтожает окно или блок безpежимного диалога и все связанные с ним дочерние окна.
* UpdateWindow – если область обновления окна непустая, то посылает сообщение wm\_Paint прямо оконной функции данного окна
* PostQuitMessage – указывает системе, что поток сделал запрос на завершение (выход).
* EndDialog – уничтожает модальное диалоговое окно, заставляя систему завершить любую обработку диалогового окна.
* PostMessage – направляет сообщение окну прикладной задачи.
* SendMessage – посылает сообщение оконной функции указанного окна. Возвpат из функции осуществляется только после обработки сообщения.
* ShowWindow – отображает или прячет окно образом, указанным параметром CmdShow.
* BeginPaint – подготавливает окно к раскраске в ответ на сообщение wm\_Paint. Заполняет Paint информацией для раскраски
* MessageBox – создает и отображает блок диалога, содержащий указанное сообщение и заголовок, а также предопределённые пиктограммы и текстовые кнопки, в соответствии с параметром TexType.
* DialogBox – создает блок модального диалога, и перед тем, как отображать диалог, посылает сообщение wm\_InitDialog.
* RegOpenKey – открывает указанный раздел реестра
* RegEnumValueW – перечисляет значения для указанного открытого раздела реестра. Функция копирует одно имя индексированного значения и блок данных для ключа при каждом вызове.
* RegDeleteValueA – удаляет указанное значение из указанного раздела реестра
* RegSetValueEx – устанавливает данные и тип указанного значения в разделе реестра.
* FindFirstFile – ищет в каталоге файл или подкаталог с именем, которое соответствует определенному имени (или частичному имени, если используются подстановочные знаки)
* FindNextFile – продолжает поиск файла от предыдущего вызова функций
* FindClose – закрывает дескриптор поиска файла, открытый функцией FindFirstFile
* GetOpenFileName – cоздает диалоговое окно «Открыть», в котором пользователь может указать диск, каталог и имя файла или набора файлов, которые нужно открыть.

1. Разработка интерфейса взаимодействия пользователя с системой.

В основном окне 3 кнопки: «Обновить», «Добавить», «Удалить»

При нажатии на кнопку «Обновить» в новом потоке запускается метод GetAllAutoStarts

При нажатии на кнопку «Удалить», после проверки на выбор строки происходит проверка метода автозапуска и в случае «Планировщика заданий» открывается диалоговое окно с выбором «отключить», «включить», «удалить», иначе программа удаляется.

При нажатии на кнопку «Добавить» открывается диалоговое окно для добавления программы в автозапуск.

В этом окне есть строка ввода пути файла для добавления и кнопка «Выбрать файл» для выбора файла с помощью проводника Windows. Также есть RadioButton c 4 позициями:

«В папку Автозагрузка», в отдельном потоке будет выполнена функция Startup::StartupAdd(text, Tools::GetStartup());

«В планировщик», в отдельном потоке будет выполнена функция Tasks::CreateTask(text);

«В реестр Учётной записи», в отдельном потоке будет выполнена функция Registry::RegAdd(HKEY\_CURRENT\_USER, text, RegRun);

«В общий реестр», в отдельном потоке будет выполнена функция Registry::RegAdd(HKEY\_LOCAL\_MACHINE, text, RegWRun).

И кнопка «Добавить» которая активирует добавление в соответствии с RadioButton.

Также в этом диалоговом окне присутствует проверка существования файла и выбора позиции RadioButton.

Для вывода списка программ, находящихся в автозапуске используется ListBox с классами:

LBS\_USETABSTOPS, для вывода данных в виде столбцов при помощи табуляции;   
LBS\_WANTKEYBOARDINPUT, для обработки сообщений из ListBox’a.

Помимо кнопок в интерфейсе присутствует обработка нажатия кнопок:   
«F5» - обновление списка программ;

«Delete» - удаление выбранной программы;

«двойное нажатие на программу в автозапуске» - удаление выбранной программы;  
переключение строк в ListBox’e с помощью стрелок в верх/в них.

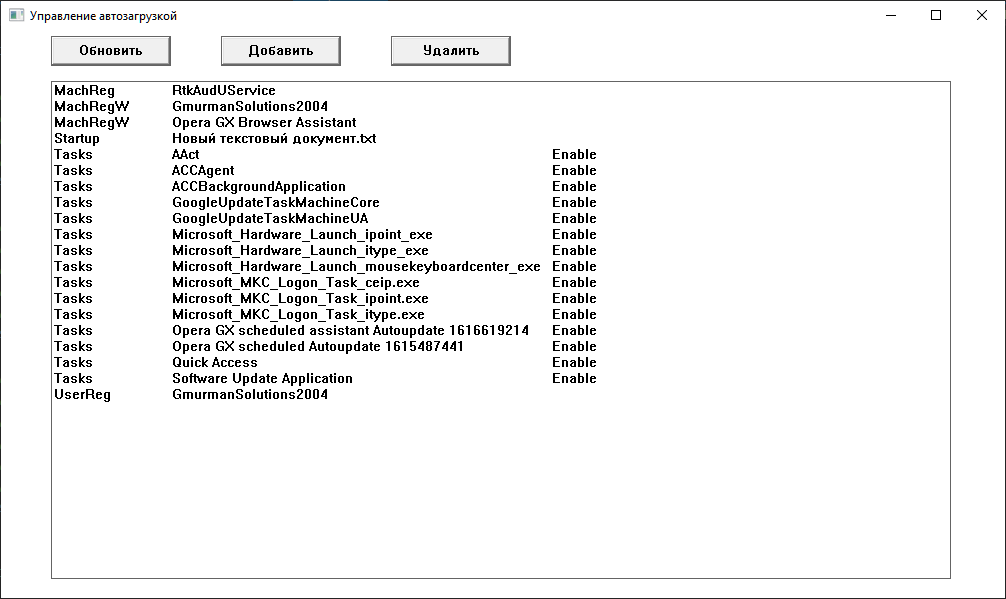


Рисунок 18. Итоговый вид главного окна.

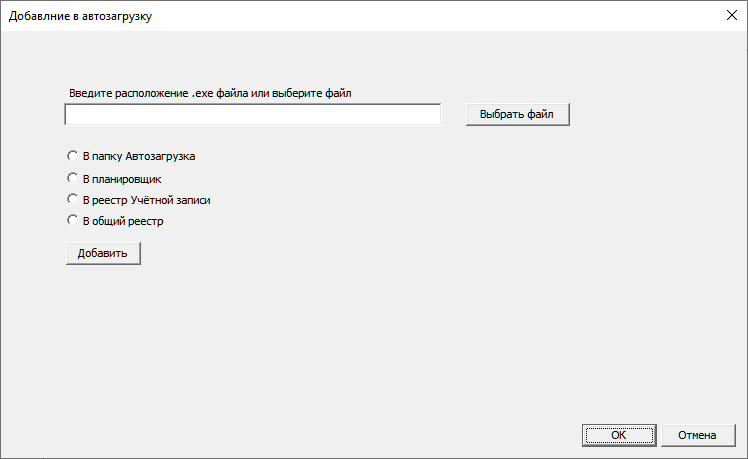


Рисунок 19. Итоговый вид окна добавления в автозагрузку.



Рисунок 20. Итоговый вид окна выбора действия.

Проектно-технологическая часть

1. Тестирование и отладка макета рабочей программы

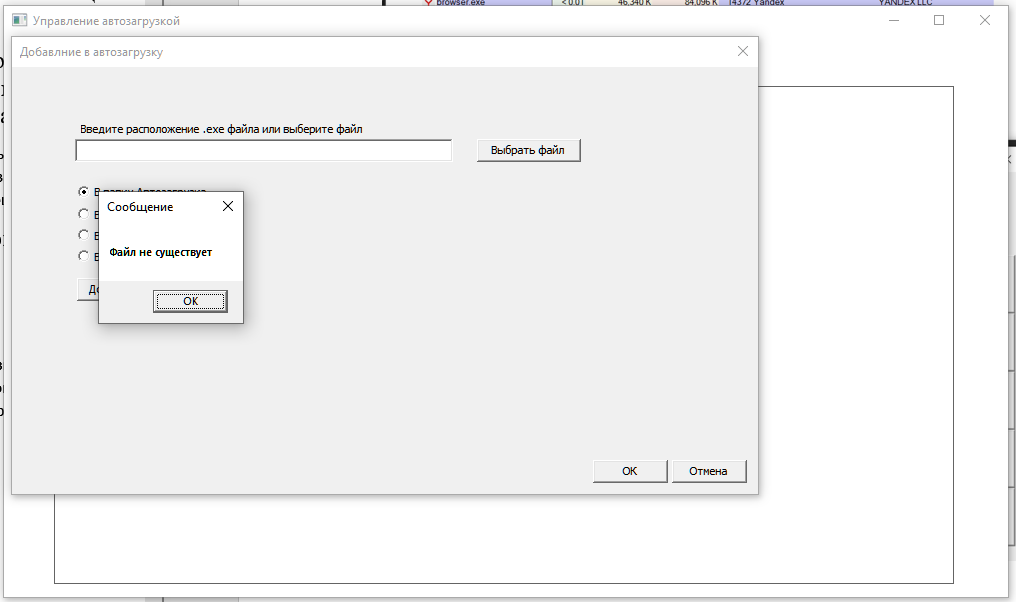
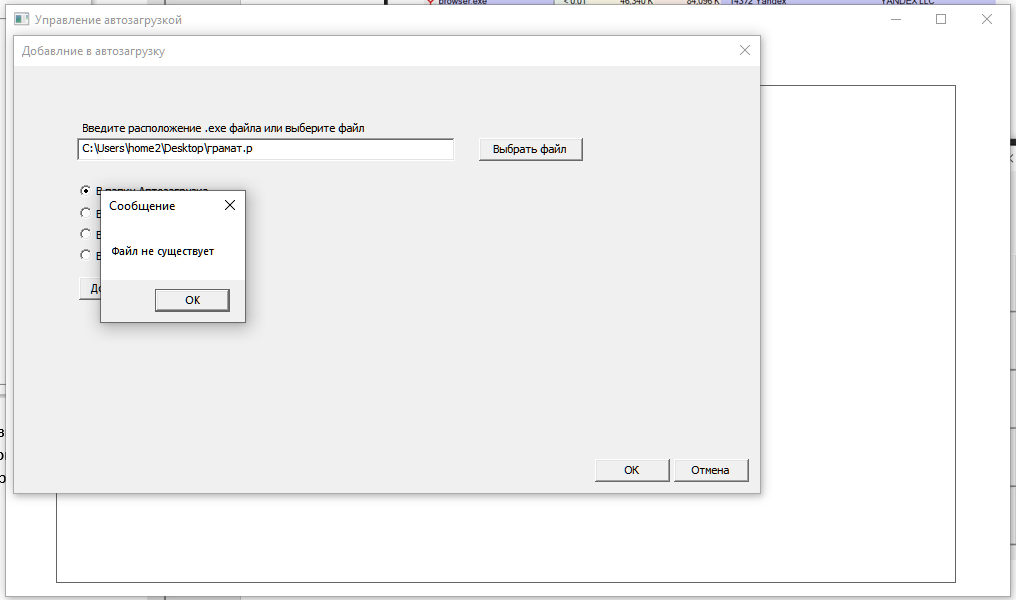


Рисунок 21. Сообщение при пустой строке расположения файла.

Рисунок 22. Сообщение при несуществующем файле.

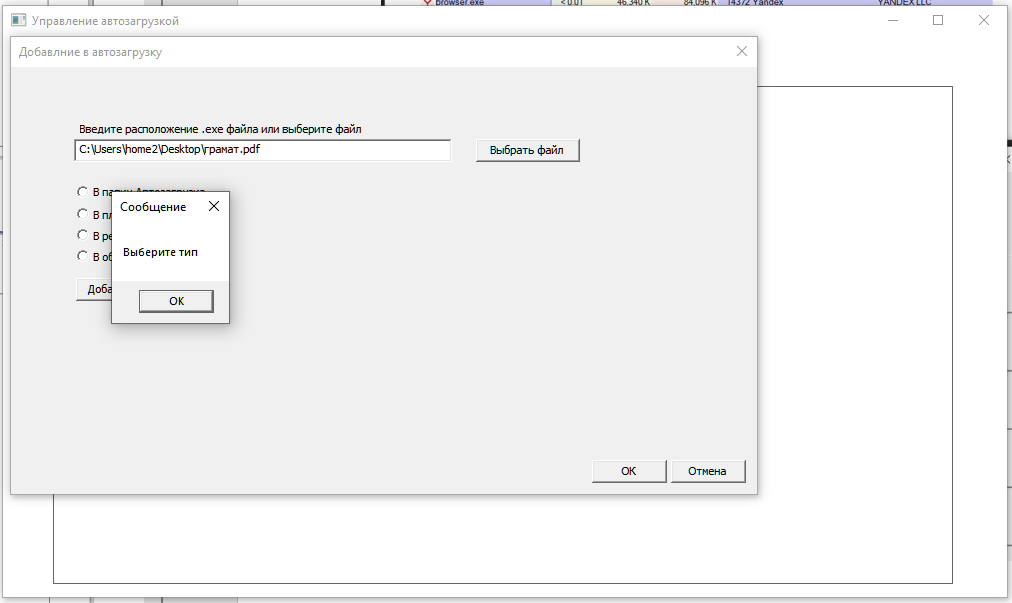


Рисунок 23. Сообщение при не выбранном методе добавления.

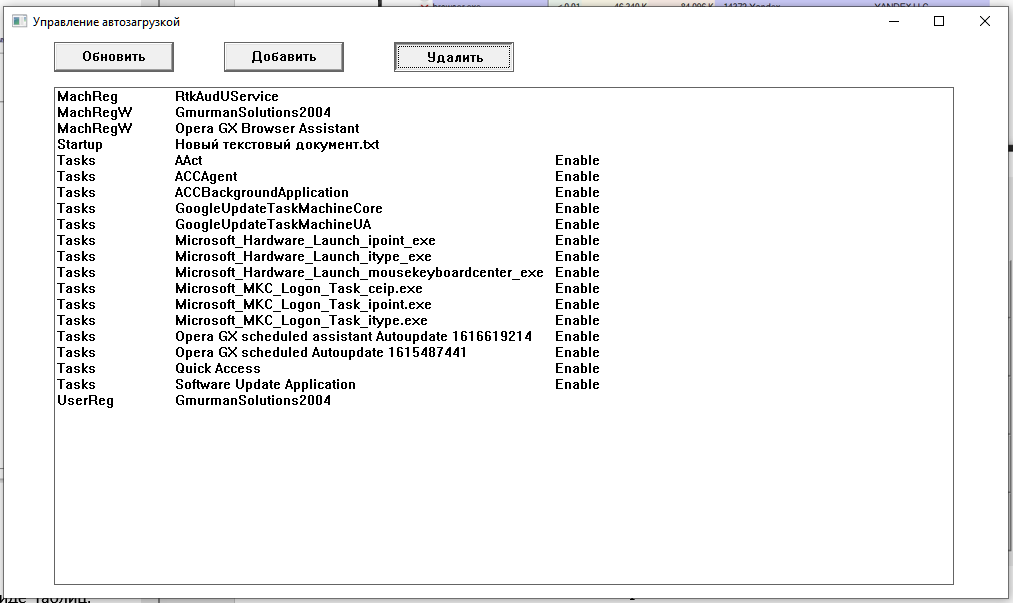


Рисунок 24. Отсутствие ошибок при нажатии кнопки «Удалить» или Delete без выбора программы.

1. Разработка руководства пользователя и руководства программиста (администратора).

Руководство администратора: для установки программы на компьютер необходимо дважды нажать на установщик «Setup.exe» и при необходимости ввести пароль. Откроется окно установщика (Рисунок 25), в этом окне нажать «Далее».

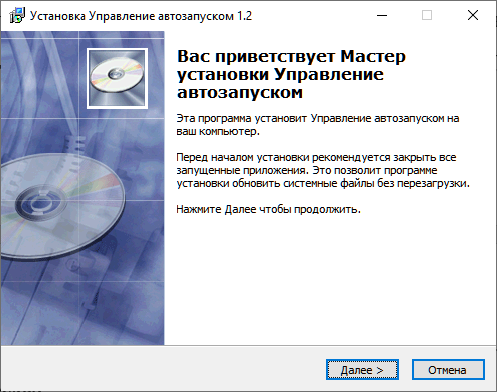


Рисунок 25. Отсутствие.

В следующем пункте нужно выбрать каталог установки и нажать «Далее» (Рисунок 26).

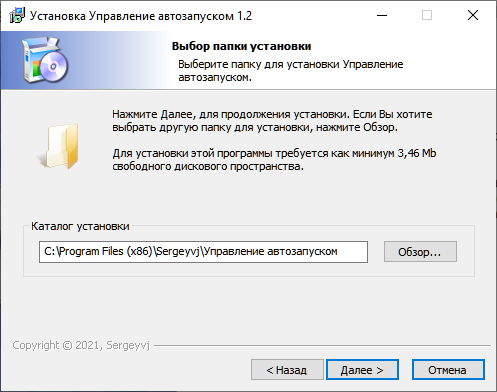


Рисунок 26. Выбор папки установки.

Рекомендуется создать ярлык на рабочем столе для удобства дальнейшего использования программы, для этого нужно проверить установлена ли галочка рядом с надписью: «Создать ярлык на рабочем столе», и нажать «Далее» для перехода к следующему пункту установки (Рисунок 27).

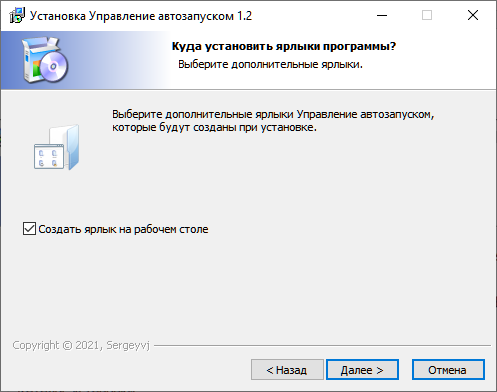


Рисунок 27. Создание ярлыка.

В следующем пункте нужно нажать «Установить» и дождаться завершения установки (Рисунок 28).

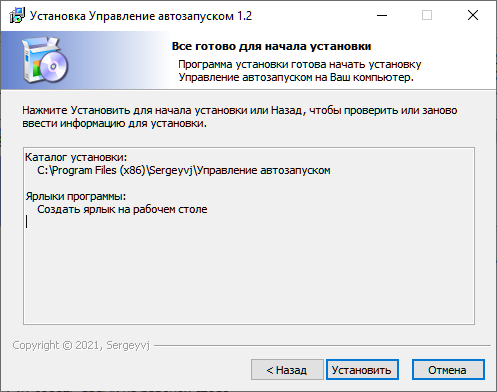


Рисунок 28. Подтверждение установки.

После завершения установки в новом пункте нужно нажать кнопку «Готово» для выхода из установщика (Рисунок 29).

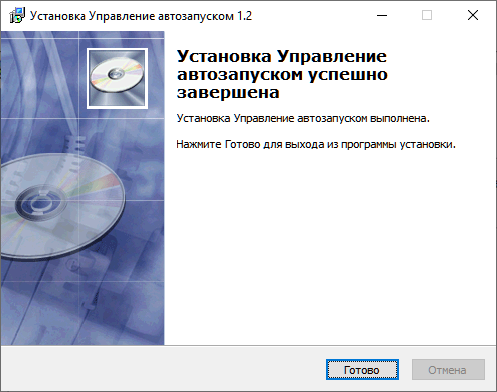


Рисунок 29. Завершение установки.

Администратор должен запустить приложение с правами администратора для корректной работы программы (Рисунок 30)



Рисунок 30. Запуск программы.

Руководство программиста: при необходимости изменения программы нужно запустить проект. Для этого необходимо установить Visual Studio 2019, открыть папку с проектом и запустить файл CourseWork.sln

Руководство пользователя: при запуске программы в основном окне отображается список программ, находящихся в автозапуске (Рисунок 31). В левом столбце указывается метод автозагрузки: Tasks – автозапуск с помощью планировщика заданий; Startup – запуск из папки «Автозагрузка»; MachReg – запуск из общего реестра; MachRegW – запуск из общего реестра; UserReg – запуск из реестра учётной записи.

В центральном столбце отображается название программы. Правый столбец заполняется только автозапуска методом планировщика заданий.

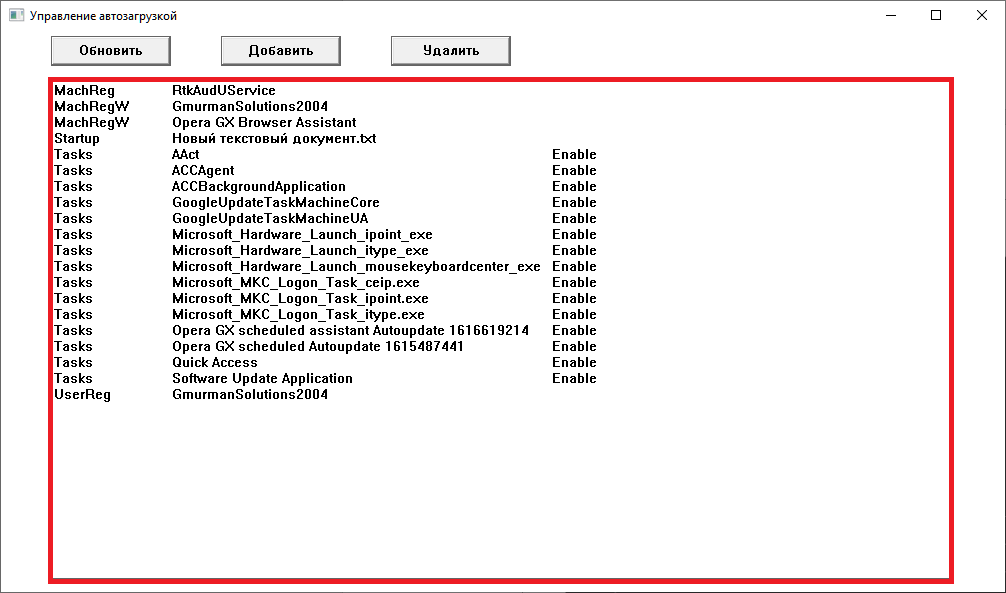


Рисунок 31. Список программ, находящихся в автозапуске.

Для удаления программы из автозапуска есть несколько вариантов: можно дважды нажать на неё; выделив нужную программу одним щелчком, нажать «Delete» на клавиатуре или кнопку «Удалить» в главном окне программы (Рисунок 32).

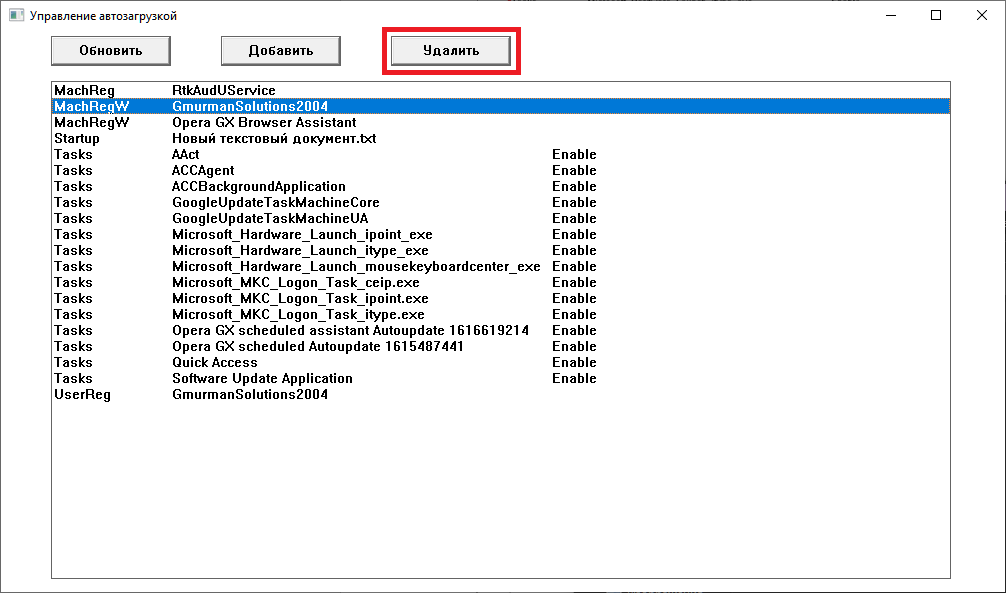


Рисунок 32. Удаление программы из автозапуска.

Для добавления программы в автозапуск нужно нажать кнопку «Добавить» (Рисунок 33). В открывшемся окне для добавления программы в автозапуск нужно выбрать файл. Для этого нужно нажать кнопку «Выбрать файл», откроется проводник для выбора файла (Рисунок 34). Затем нужно выбрать метод добавления в автозапуск и нажать кнопку «Добавить» (Рисунок 35).

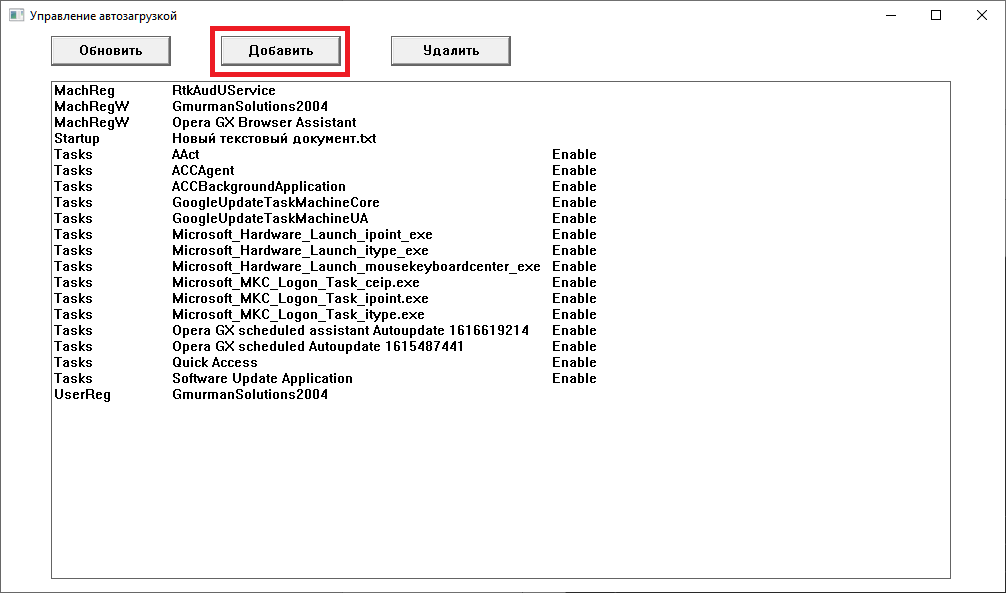


Рисунок 33. Кнопка для добавления программы.

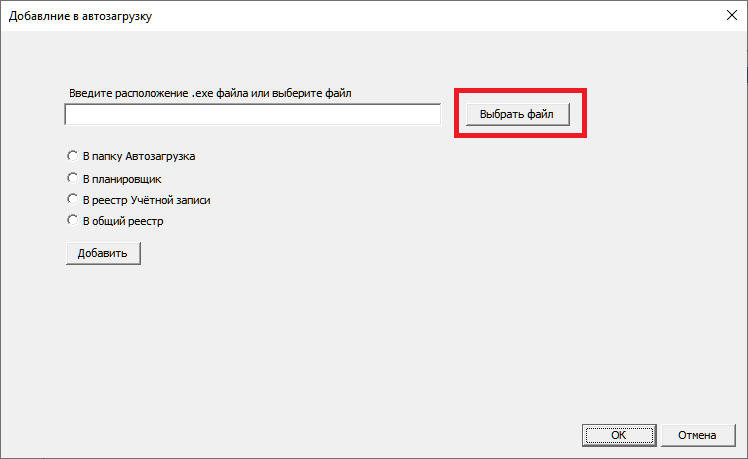


Рисунок 34. Выбор файла.

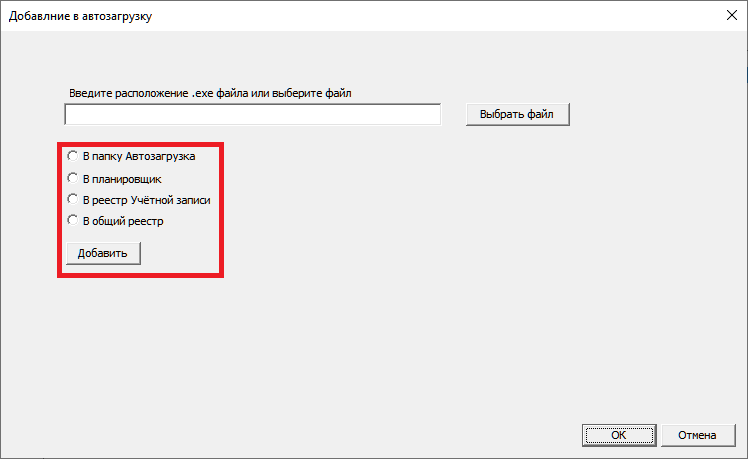


Рисунок 35. Выбор метода автозапуска.

Также при необходимости можно обновить список программ, находящихся в автозапуске, для этого нужно нажать кнопку «Обновить» (Рисунок 36) или нажать клавишу «F5» на клавиатуре.

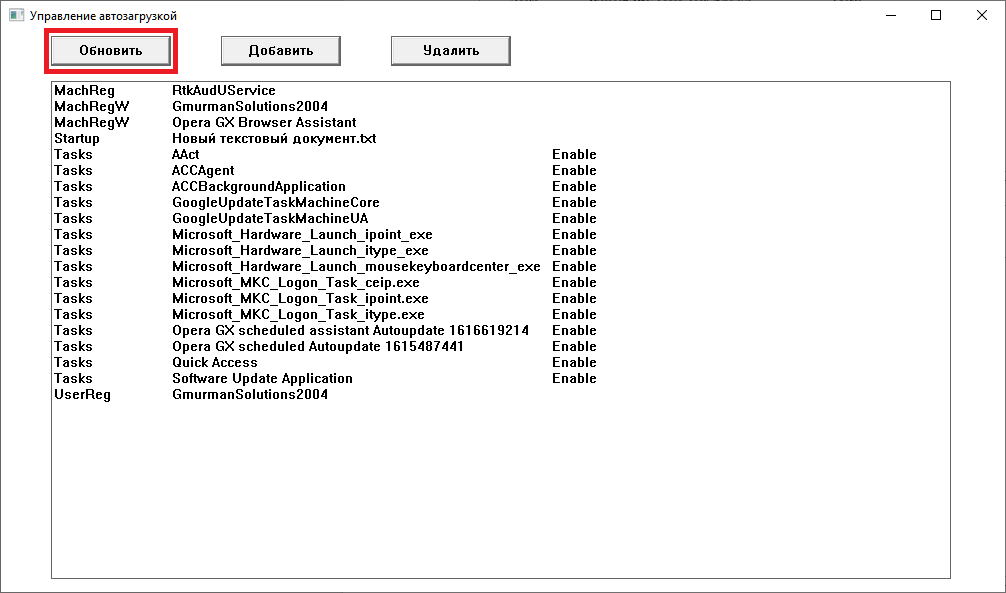


Рисунок 36. Кнопка обновления списка.

Источники разработки

1. Автозагрузка программ Windows 10  
   <https://remontka.pro/autorun-windows-10/>
2. ГОСТ 34.602-89 Техническое задание на создание АС.  
   <https://www.mos.ru/dit/documents/tekhnicheskaia-politika/view/317220/>
3. Краткий справочник по функциям WinAPI  
   <http://www.codenet.ru/progr/delphi/winapi/>
4. Литвиненко Н.А. Технология программирования на С++. Win32 API-приложения / БХВ-Петербург, 2010. – 288 с.
5. Павловская Т.А. C/C++ . Процедурное и объектно-ориентированное программирование. Стандарт 3-го поколения. СПб.: Питер, 2015.
6. Пошаговое руководство. создание традиционного классического приложения Windows (C++)  
   <https://docs.microsoft.com/ru-ru/cpp/windows/walkthrough-creating-windows-desktop-applications-cpp?view=msvc-160&viewFallbackFrom=vs-2019>
7. Создание элементов интерфейса и реализация функционала приложения использующего Win API  
   <https://docs.microsoft.com/ru-ru/windows/win32/apiindex/windows-api-list>