**СОДЕРЖАНИЕ**

1 Общие сведения 6

1.1. Техническое задание. 6

1.1.1 Полное наименование системы и ее условное обозначение. 6

1.1.2 Шифр темы или шифр (номер) договора. 6

1.1.3 Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты. 6

1.1.4 Перечень документов, на основании которых создается система, кем и когда утверждены эти документы. 6

1.1.5 Плановые сроки начала и окончания работы по созданию системы. 6

1.1.6 Сведения об источниках и порядке финансирования работ. 7

1.2. Назначение и цели создания (развития) системы. 7

1.2.1 Назначение системы. 7

1.2.2 Цели создания системы. 7

1.3. Характеристика объектов автоматизации. 7

1.3.1 Краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию. 7

1.4. Требования к системе. 7

1.4.1 Требования к структуре и функционированию системы. 8

1.4.2 Требования к защите информации от несанкционированного доступа. 8

1.4.3 Требования к входным/выходным данным. 8

1.4.4 Требования к составу и параметрам технических средств. 8

1.5. Состав и содержание работ по создание системы 9

1.6. Порядок контроля и приемки системы. 9

1.6.1 Виды испытаний 9

1.6.2 Общие требования к приемке работ 10

1.7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие 10

1.8. Требование к документированию 10

1.9. Источники разработки 10

2 Научно-исследовательская часть 11

2.1 Постановка задачи проектирования 11

2.2 Описание предметной области 11

2.3 Аналоги разрабатываемого приложения: 11

2.4 Перечень задач, подлежащих решению в процессе разработки 13

2.5 Обоснование выбора инструментов и платформы для разработки. 14

2.6 Платформа для разработки. 16

3 Проектно-конструкторская часть 18

3.1 Разработка структуры приложения 18

3.2 Используемые функции Win32API 18

3.3 Разработка алгоритмов обработки информации. 26

3.4 Разработка архитектуры приложения. 28

3.5 Разработка граф интерфейса взаимодействия пользователя с системой 30

4 Проектно-технологическая часть 33

4.1 Тестирование функциональных задач 33

4.2 Руководство пользователя 35

Заключение 36

Список источников 37

Приложение………………………………………………………………………. 38

# Общие сведения

# Техническое задание

# Полное наименование системы и ее условное обозначение

Текстовый редактор с расширенными возможностями на основе Windows API.

# Шифр темы или шифр (номер) договора

Шифр темы и договора отсутствует.

# Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты

Высшее учебное заведение КФ МГТУ имени Н. Э. Баумана.

# Перечень документов, на основании которых создается система, кем и когда утверждены эти документы

На основании задания курсовой работы по учебному плану.

# Плановые сроки начала и окончания работы по созданию системы

Начало работы: 15.02.2021

Окончание работы: 10.05.2021

# Сведения об источниках и порядке финансирования работ

Проект осуществляется без финансирования.

# Назначение и цели создания (развития) системы

# 1.2.1 Назначение системы

Работа с текстовыми файлами.

# 1.2.2 Цели создания системы

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

Целью разработки является создание утилиты или программного обеспечения, позволяющего оперативно и в удобной форме просматривать и редактировать текстовые файлы.

# Характеристика объектов автоматизации

# 1.3.1 Краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию

Объектом автоматизации является набор функций, обеспечивающий создание и изменение текстовых файлов.

# Требования к системе

# 1.4.1 Требования к структуре и функционированию системы

Предназначено для создания и изменения текстовых файлов.

# 1.4.2 Требования к защите информации от несанкционированного доступа

Требования к защите информации от несанкционированного доступа, отсутствуют.

# 1.4.3 Требования к входным/выходным данным

Входные данные:

текстовый файл .txt .

Выходные данные:

после редактирования добавляет время, когда файл был записан.

# 1.4.4 Требования к составу и параметрам технических средств

Для использования программы необходимы средства ввода и вывода информации:

мышь;

клавиатура;

монитор.

# 1.5. Состав и содержание работ по создание системы

|  |  |
| --- | --- |
| Стадии | Этапы работ |
| 1. Формирование требований к АС | 1.1. Обследование объекта и обоснование необходимости создания АС.  1.2. Формирование требований пользователя к АС.  1.3. Оформление отчёта о выполненной работе и заявки на разработку АС (тактико-технического задания) |
| 2. Разработка концепции АС. | 2.1. Изучение объекта.  2.2. Проведение необходимых научно-исследовательских работ.  2.3. Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя.  2.4. Оформление отчёта о выполненной работе. |
| 3. Техническое задание. | Разработка и утверждение технического задания на создание АС. |
| 4. Разработка системы. | Разработка системы согласно техническому заданию. |
| 5. Защита курсовой работы. | 5.1. Подготовка презентации и речи для защиты курсовой работы.  5.2. Защита курсовой работы. |

# 

# 1.6. Порядок контроля и приемки системы.

# 1.6.1 Виды испытаний

Тесты на работоспособность программы с помощью средств тестирования Microsoft Visual Studio.

# 1.6.2 Общие требования к приемке работ

При проведении приемки работ должна быть произведена проверка выполненного задания на соответствие требованиям текущего «Технического задания».

# 1.7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Для работы программного продукта требуется запустить исполняемый файл.

# 1.8. Требование к документированию

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:

структурные схемы;

алгоритмические схемы.

# 1.9. Источники разработки

1. Е.В. Вершинин, Р.Б. Бобров “Методические указания к выполнению курсовой работы” Калуга КФ МГТУ им. Баумана 2021.
2. 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>

# Научно-исследовательская часть

# Постановка задачи проектирования

Задача курсовой работы состоит в разработке программы – текстового редактора с расширенными возможностями на основе Windows API, которая позволяет после конца редактирования текста добавляет время когда файл был записан.

Необходимо разработать удобный пользовательский интерфейс для работы с программой.

# Описание предметной области

Предметной областью является текстовый редактор с расширенным возможностями. Задача которого чтение, редактирование и запись в тот же или другой формат. Он должен быть удобным и простым в использовании. А так же иметь необычные функции, например:

* после конца редактирования текста добавляет время, когда файл был записан.

# 2.3 Аналоги разрабатываемого приложения

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

Для анализа следует начать с встроенной программы “Блокнот”.  
**Блокнот** — простой текстовый редактор, являющийся частью операционных систем Microsoft Windows, начиная с вышедшей в 1985 году Windows 1.0.  
Он по сути не имеет расширенного функционала, а так же отсутствует возможность записи в файл времени когда пользователь его редактировал.

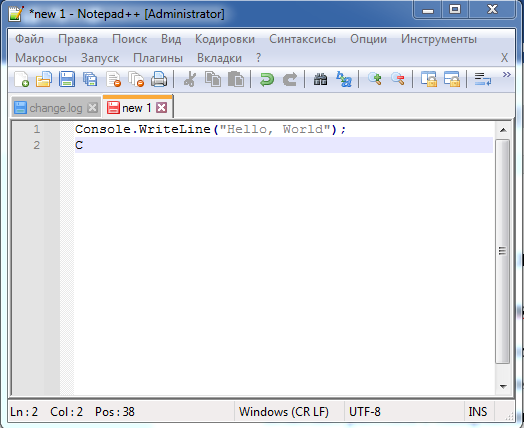


Рисунок 1

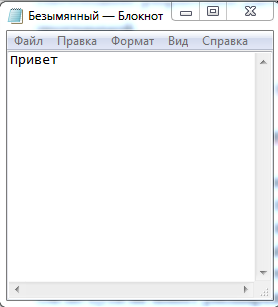


Рисунок 2

Рассмотрим **Notepad++**Возможности:

- Подсветка кодов в редакторе;

- Подсветка синтаксиса;

- Автоматическое дописывание тегов;

- Постановка маркеров в редактируемом коде.

А так же можно использовать дополнительные плагины для расширения возможностей.

# 2. 4 Перечень задач, подлежащих решению в процессе разработки

Задачи, подлежащие решению в процессе разработки, следующие:

1) Программа должна по умолчанию читать разные форматы файлов;

2) Программа должна иметь простой, но в то же время понятный и наглядный интерфейс, который не должен перегружать ресурсы компьютера;

3) программа должна иметь возможность после конца редактирования текста добавляет время когда файл был записан;

4) пользователь должен иметь возможность не только читать содержимое, но и редактировать его;

5) программа не должна занимать большой объем памяти;

6) работоспособность приложения в среде Windows.

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.

WinAPI спроектирован для использования в языке С++ для написания прикладных программ, предназначенных для работы под управлением операционной системы MS Windows.

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

Язык С++ является одним из самых распространенных языков.

# 2.5 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 млрд 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.

По статистике ОС для компьютеров операционная система Windows заняла первое место. Ее распространение составило 76,58% от количества всех ОС.

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.

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.

# 2.6 Платформа для разработки.

Microsoft Visual Studio — линейка продуктов компании Microsift включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения так и игры и приложения с графическим интерфейсом в том числе с поддержкой технологии Windows form а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, WindowsMobile, WindowsCE, .NETFramework, Xbox, WindowsPhone .NETCompactFramework и Silverlight.

Visual Studio включает в себя редакторисходногокода с поддержкой технологии IntelliSense и возможностью простейшего рефакторингакода. 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 схемыбазыданных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроляверсийисходногокода (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).

# Проектно-конструкторская часть

# Разработка структуры приложения

Для разработки приложения использовалась среда Microsoft Visual Studio, включающая в себя широкий набор функциональных инструментов. Для представления принципа работы программы ниже приведён демонстрационный чертёж.

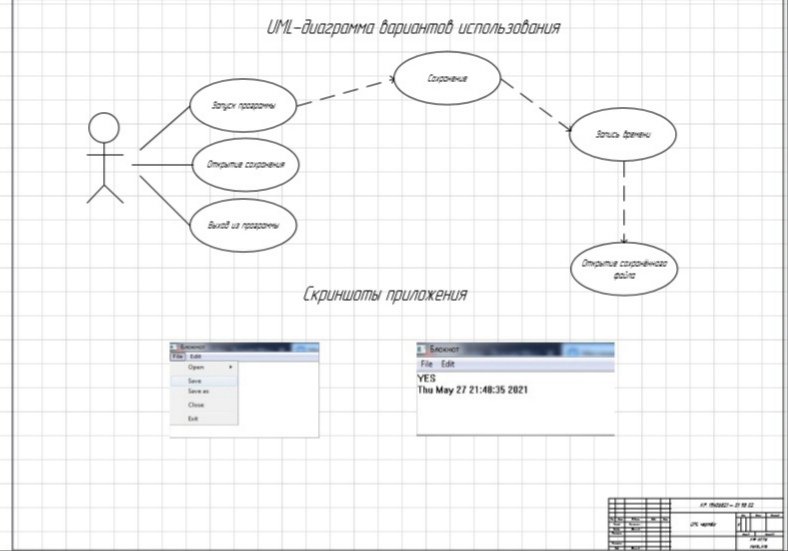


Рисунок 3

### Используемые функции Win32API

LRESULT CALLBACK MainWindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);  
  
HWND hEdit;  
  
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {  
HINSTANCE g\_hInst = hInst;  
WNDCLASSW wc = { 0 };  
wc.hbrBackground = (HBRUSH)COLOR\_WINDOW;  
wc.hCursor = LoadCursor(NULL, IDC\_ARROW);  
wc.hInstance = hInst;  
wc.lpszClassName = L"MainWndClass";  
wc.lpfnWndProc = MainWindowProc;

Здесь объявляется сигнатура главной процедуры окна (та, которая будет обрабатывать входящие сообщения), затем создаётся переменная для хранения хендла поля ввода текста. Хендл - это подобие указателя, через него можно взаимодействовать с окном, к которому он привязан

WinMain - это точка входа в программу, тут создаётся структура WNDCLASSW и мы заполняем её данными, необходимыми для регистрации класса. Регистрация - это, фактически, создание нового класса окна, экземпляры которого потом уже можно будет создавать. Тут задаётся название, цвет фона, курсор и указатель на процедуру окна.

if (!RegisterClassW(&wc))  
return -1;  
  
CreateWindowW(  
L"MainWndClass", L"Блокнот",  
WS\_OVERLAPPEDWINDOW | WS\_VISIBLE,  
400, 200, 500, 250,  
NULL, NULL, NULL, NULL  
);  
  
MSG msg = { 0 };  
while (GetMessage(&msg, NULL, NULL, NULL)) {  
TranslateMessage(&msg);  
DispatchMessage(&msg);  
}  
return 0;  
}

Здесь мы именно регистрируем класс и проверяем, получилось ли его создать. Если не получилось, программа завершается с кодом -1, если получилось - мы создаём экземпляр главного окна (CreateWindow). Первый аргумент - название класса, второй - заголовок окна, дальше идёт стиль окна, его положение и размеры. Последние аргументы - родитель (окно новое, родителя нет), меню (пока никакое), экземпляр приложения (нулл) и дополнительные параметры (никаких, null).

Затем создаётся структура сообщения и запускается главный цикл приложения, который вытаскивает сообщения из очереди (GetMessage, первый аргумент - указатель на структуру для записи сообщения, второй - окно, от которого нужны сообщения, если null - все от текущего приложения, затем фильтры, оба нулл - все сообщения), разбирает их (TranslateMessage) и доставляет в обработку основной процедуре (Dispatch Message).

void Editor(HWND hWnd)  
{  
hEdit = CreateWindowW(L"Edit", L"", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | WS\_BORDER | WS\_VSCROLL | WS\_HSCROLL,  
10, 50, 400, 300, hWnd, NULL, NULL, NULL);  
}

Создаём экземпляр поля ввода текста (класс Edit).

void Menubar(HWND hWnd) {  
HMENU MainMenu = CreateMenu();  
HMENU FileMenu = CreateMenu();  
HMENU OpenMenu = CreateMenu();  
HMENU EditMenu = CreateMenu();  
  
AppendMenuW(MainMenu, MF\_POPUP, (UINT\_PTR)FileMenu, L"File");  
AppendMenuW(FileMenu, MF\_POPUP, (UINT\_PTR)OpenMenu, L"Open");  
AppendMenuW(OpenMenu, MF\_STRING, MB\_TXT, L".txt");  
AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(FileMenu, MF\_STRING, MB\_SAVE, L"Save");  
AppendMenuW(FileMenu, MF\_STRING, MB\_SAVEAS, L"Save as");  
AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(FileMenu, MF\_STRING, MB\_CLOSE, L"Close");  
AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(FileMenu, MF\_STRING, MB\_EXIT, L"Exit");  
  
AppendMenuW(MainMenu, MF\_POPUP, (UINT\_PTR)EditMenu, L"Edit");  
AppendMenuW(EditMenu, MF\_STRING, MB\_CUT, L"Cut");  
AppendMenuW(EditMenu, MF\_STRING, MB\_COPY, L"Copy");  
AppendMenuW(EditMenu, MF\_STRING, MB\_PASTE, L"Paste");  
AppendMenuW(EditMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(EditMenu, MF\_STRING, MB\_SELALL, L"Select all");  
  
SetMenu(hWnd, MainMenu);  
}

Здесь мы посредством Create Menu создаём четыре меню, три из которых мы выкладываем в главное. Главное меню - это сама полоска вверху приложения, вложенные - это file и edit в главном и подменю open в file. Через Append Menu, собственно, мы их друг в друга и выкладываем. MF\_SEPARATOR - это полоска - разделитель в меню. В конце мы через SetMenu назначаешь получившееся меню главному окну.

void DisplayTXT(wchar\_t\* path)  
{  
  
FILE\* file = \_wfopen(path, L"rt");  
if (file == NULL)  
{  
return;  
}  
  
fseek(file, 0, SEEK\_END);  
int \_size = ftell(file) / sizeof(wchar\_t);  
rewind(file);  
  
wchar\_t\* data = new wchar\_t[\_size + 1];  
fread(data, sizeof(wchar\_t), \_size, file);  
data[\_size] = '\0';  
  
SetWindowTextW(hEdit, data);  
delete[] data;  
  
fclose(file);  
}

Функция, которая открывает файл в формате UTF-16, считывает его и переносит содержимое в текстовое поле.

void OpenTXT(HWND hWnd)  
{  
OPENFILENAMEW ofn;  
  
wchar\_t file\_name[100];  
  
ZeroMemory(&ofn, sizeof(ofn));  
ofn.lStructSize = sizeof(ofn);  
ofn.hwndOwner = hWnd;  
ofn.lpstrFile = file\_name;  
ofn.lpstrFile[0] = L'\0';  
ofn.nMaxFile = MAX\_PATH;  
ofn.lpstrFilter = L"txt Files\00\*.TXT\0";  
ofn.nFilterIndex = 1;  
  
GetOpenFileNameW(&ofn);  
  
DisplayTXT(ofn.lpstrFile);  
}

Эта функция создаёт диалоговое окно через GetOpenFileName, которое даёт нам выбрать файл, и потом вызывает прошлую функцию с полученным путем до файла.

void WriteTXT(wchar\_t\* path)  
{  
FILE\* file = \_wfopen(path, L"w+t");  
int \_size = GetWindowTextLengthW(hEdit);  
  
LPWSTR data = new WCHAR[\_size+1];  
GetWindowTextW(hEdit, data, \_size+1);  
data[\_size] = L'\0';  
  
fwrite(data, sizeof(wchar\_t), \_size, file);  
delete[] data;  
  
time\_t t;  
time(&t);  
wchar\_t \*str = \_wctime(&t);  
fwrite(str, sizeof(wchar\_t), wcslen(str), file);  
delete[] str;  
  
fclose(file);  
}

Здесь происходит открытие файла на запись и запись в него содержимого из текстового поля. Потом мы получаем текущее время и дополняешь им файл.

void SavetoTXT(HWND hWnd)  
{  
OPENFILENAMEW ofn;  
  
wchar\_t file\_name[100];  
  
ZeroMemory(&ofn, sizeof(ofn));  
  
ofn.lStructSize = sizeof(ofn);  
ofn.hwndOwner = hWnd;  
ofn.lpstrFile = file\_name;  
ofn.lpstrFile[0] = L'\0';  
ofn.nMaxFile = MAX\_PATH;  
ofn.lpstrFilter = L"txt Files\00\*.TXT\0";  
ofn.nFilterIndex = 1;  
  
GetSaveFileNameW(&ofn);  
  
WriteTXT(ofn.lpstrFile);  
}

Здесь мы аналогично открытию файла получаем адрес, по которому файл нужно сохранить, и вызываем прошлую процедуру.

LRESULT CALLBACK MainWindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {  
RECT rct;  
switch (msg)  
{  
case WM\_CREATE: {  
Menubar(hWnd);  
Editor(hWnd);  
break;  
}  
case WM\_COMMAND: {  
switch (wp) {  
case MB\_TXT: {  
OpenTXT(hWnd);  
break;  
}  
case MB\_SAVE: {  
SavetoTXT(hWnd);  
break;  
}  
case MB\_CUT: {  
SendMessage(hEdit, WM\_CUT, 0, 0);  
break;  
}  
case MB\_PASTE: {  
SendMessage(hEdit, WM\_PASTE, 0, 0);  
break;  
}  
case MB\_COPY: {  
SendMessage(hEdit, WM\_COPY, 0, 0);  
break;  
}  
case MB\_SELALL: {  
SendMessage(hEdit, EM\_SETSEL, 0, -1);  
break;  
}  
case MB\_EXIT: {  
PostQuitMessage(0);  
break;  
}  
}  
break;  
}  
case WM\_CLOSE: {  
PostQuitMessage(0);  
break;  
}  
case WM\_SIZE: {  
MoveWindow(hEdit, 0, 0, LOWORD(lp), HIWORD(lp), FALSE);  
break;  
}  
}  
return DefWindowProcW(hWnd, msg, wp, lp);  
}

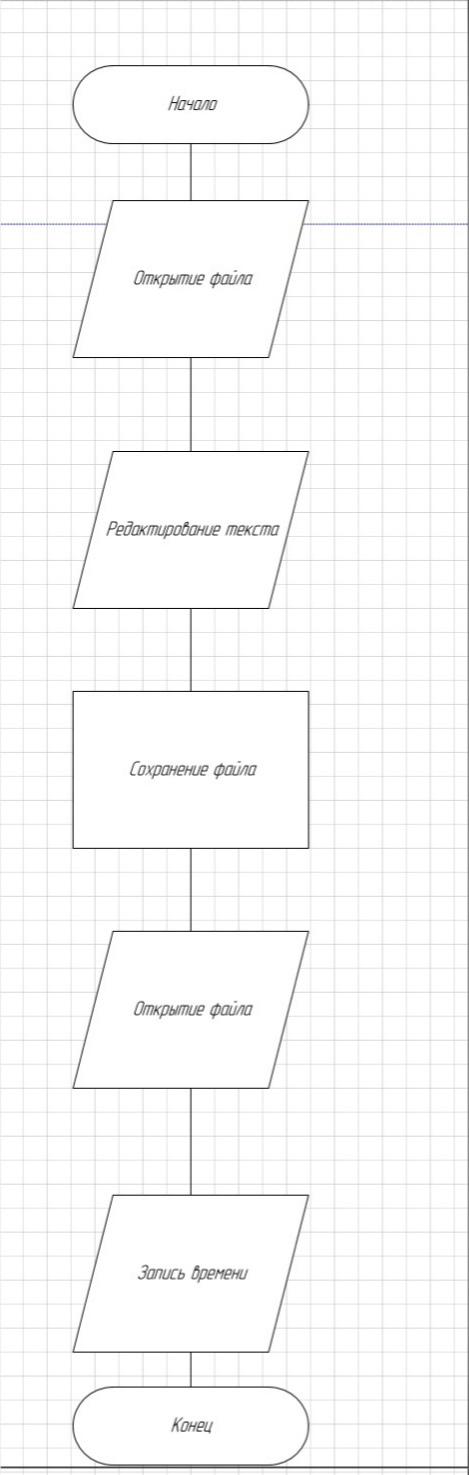
Это реализация главной процедуры окна, если оно только создано - мы вызываем методы для его наполнения, если была нажата кнопка в меню - мы выполняем соответствующее действие, если выход или закрытие окна - закрываем приложение, если смена размера - мы меняем размер текстового поля. В конце мы вызываем стандартную процедуру окна, которая осуществляет действия по умолчанию.

Для работы каждого пункта меню – необходимо присвоить ID.

[#pragma](https://vk.com/im?sel=560282801&st=%23pragma) once  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_TXT 100  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_SAVE 101  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_SAVEAS 102  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_CLOSE 103  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_EXIT 104  
  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_CUT 112  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_COPY 113  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_PASTE 114  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_SELALL 115  
  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_OPEN 10  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_SAVE 11  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_SAVEAS 12  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_EXIT 13  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_ABOUT 14  
  
[#define](https://vk.com/im?sel=560282801&st=%23define) ID\_EDIT 20  
[#define](https://vk.com/im?sel=560282801&st=%23define) ID\_STATUS 21

# Разработка алгоритмов обработки информации.

При открытии ранее сохранённого файла пользователь может видеть время когда файл закончили редактировать.

Рисунок 4

Так же пользователь может работать с окном выполняя там разные функции.

# Рисунок 5

# Разработка архитектуры приложения.

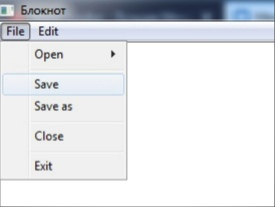


Рисунок 6

Пользовательский интерфейс. Для работы каждого пункта меню – необходимо присвоить ID. Далее представлен листинг подключаемого заголовка с инициализацией этих ID.

[#pragma](https://vk.com/im?sel=560282801&st=%23pragma) once  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_TXT 100  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_SAVE 101  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_SAVEAS 102  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_CLOSE 103  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_EXIT 104  
  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_CUT 112  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_COPY 113  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_PASTE 114  
[#define](https://vk.com/im?sel=560282801&st=%23define) MB\_SELALL 115  
  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_OPEN 10  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_SAVE 11  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_SAVEAS 12  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_EXIT 13  
[#define](https://vk.com/im?sel=560282801&st=%23define) IDC\_ABOUT 14  
  
[#define](https://vk.com/im?sel=560282801&st=%23define) ID\_EDIT 20  
[#define](https://vk.com/im?sel=560282801&st=%23define) ID\_STATUS 21

После присвоения ID можем создавать само меню. Листинг создания меню представлен ниже:

void Menubar(HWND hWnd) {  
HMENU MainMenu = CreateMenu();  
HMENU FileMenu = CreateMenu();  
HMENU OpenMenu = CreateMenu();  
HMENU EditMenu = CreateMenu();  
  
AppendMenuW(MainMenu, MF\_POPUP, (UINT\_PTR)FileMenu, L"File");  
AppendMenuW(FileMenu, MF\_POPUP, (UINT\_PTR)OpenMenu, L"Open");  
AppendMenuW(OpenMenu, MF\_STRING, MB\_TXT, L".txt");  
AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(FileMenu, MF\_STRING, MB\_SAVE, L"Save");  
AppendMenuW(FileMenu, MF\_STRING, MB\_SAVEAS, L"Save as");  
AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(FileMenu, MF\_STRING, MB\_CLOSE, L"Close");  
AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(FileMenu, MF\_STRING, MB\_EXIT, L"Exit");  
  
AppendMenuW(MainMenu, MF\_POPUP, (UINT\_PTR)EditMenu, L"Edit");  
AppendMenuW(EditMenu, MF\_STRING, MB\_CUT, L"Cut");  
AppendMenuW(EditMenu, MF\_STRING, MB\_COPY, L"Copy");  
AppendMenuW(EditMenu, MF\_STRING, MB\_PASTE, L"Paste");  
AppendMenuW(EditMenu, MF\_SEPARATOR, NULL, NULL);  
AppendMenuW(EditMenu, MF\_STRING, MB\_SELALL, L"Select all");  
  
SetMenu(hWnd, MainMenu);  
}

# Разработка граф интерфейса взаимодействия пользователя с системой

Нажимаем File Open выбирает формат файла .txt.

Рисунок 7

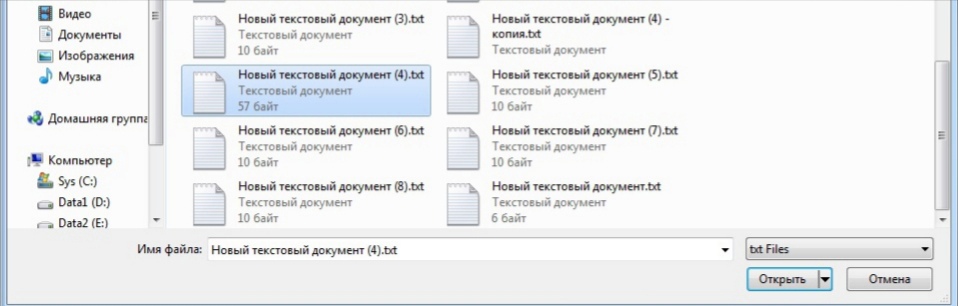
Выбираем нужный нам файл.

Рисунок 8

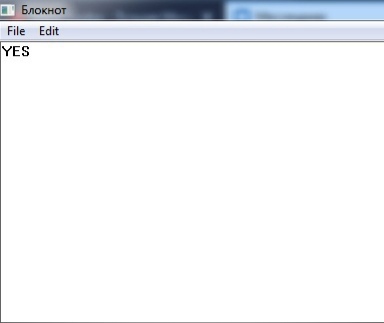
Редактируем файл.

Рисунок 9

Сохраняем файл.

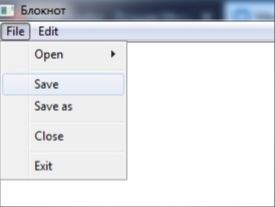
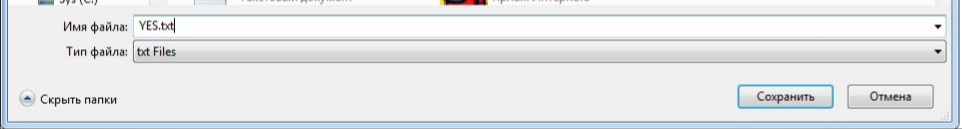


Рисунок 10

Рисунок 11

Видим то время когда мы в прошлый раз закончили редактировать файл.

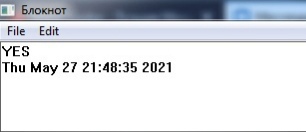


Рисунок 12

Для составления интерфейса в проект были добавлены следующие компоненты:

Окно приложения;

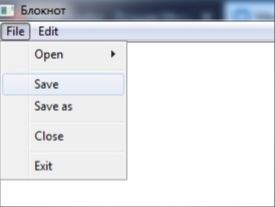
Окно редактирования;

Меню.

# Проектно-технологическая часть

# 4.1 Тестирование функциональных задач

Главной функцией программы является то что при закрытии после конца редактирования файла добавляется время когда файл был записан.

Рисунок 13

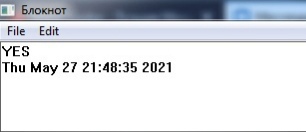


Рисунок 14

Так же немаловажна функция редактирования файла.

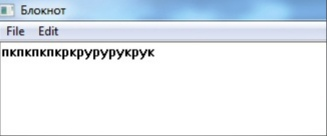


Рисунок 15. Набор текста

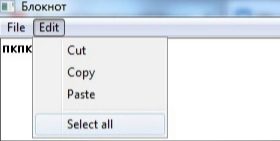


Рисунок 16. Нажатие кнопки Select all

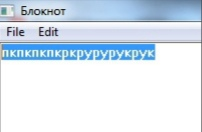


Рисунок 17. Выделение текста

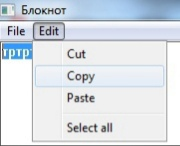


Рисунок 18. Нажатие кнопки Copy

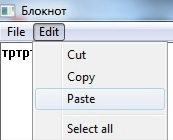


Рисунок 19. Выделение кнопки Paste

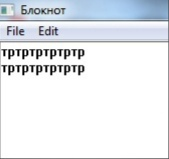


Рисунок 20. Вставленный текст

# 4.2 Руководство пользователя

Запуск программы осуществляется по открытию файла Project1.exe. После запуска открывается главное окно приложения

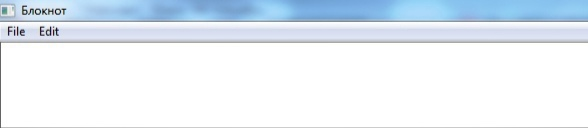


Рисунок 21

# Входные данные представлены в виде программного кода, который необходимо выполнить при определенных действиях пользователя, а именно:

# нажатие клавиш клавиатуры;

# работа пользователя по выбору и открытию файла.

# Перечень элементов интерфейса:

“File” – меню работы с файлами;

“Edit” – меню работы с текстом;

“Open” – открытие файлов;

“Save” – сохранение и окна реактирования;

“Cut” – вырезать;

“Copy” – копировать;

“Paste” – вставить;

“Select all” – выделить весь текст;

“Exit”

# Заключение

В ходе выполнения курсовой работы была выполнена задача проектирования по формированию практических навыков по разработке и реализации текстового редактора с расширенными возможностями. Также были исследованы основные функции WinAPI для работы над текстовыми файлами.В курсовой работе было разработано программное обеспечение, позволяющее оперативно и в удобной форме просматривать и изменять текст.

# Список источников

1. Сидоров В.Н., Сломинская Е.Н., Полникова Т.В., Макарова О.Ю. Оформление графической части выпускной квалификационной работы. Учебное пособие. М.: МГТУ им. Н.Э. Баумана, 2016.

2. 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.

3. Создание элементов интерфейса и реализация функционала приложения, использующего Win API <https://docs.microsoft.com/ru-ru/windows/win32/apiindex/windows-api-list>.  
4. Текстовый редактор WORD 2000. Часть 1: лабораторный практикум https://e.lanbook.com/book/116762.

5. Текстовый редактор WORD 2000. Часть 2: лабораторный практикум <https://e.lanbook.com/book/116763>.  
6. Проектирование цифровых устройств https://e.lanbook.com/book/168881.

7. Методика обучения информатике <https://e.lanbook.com/book/139269>.

**Приложение**

Source.cpp

#include <Windows.h>

#include <string.h>

#include <wchar.h>

#include <cstdio>

#include "Header.h"

#include <iostream>

#include <fstream>

#include <iterator>

#include <vector>

#include "commdlg.h"

#pragma warning (disable:4996)

LRESULT CALLBACK MainWindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

HWND hEdit;

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

HINSTANCE g\_hInst = hInst;

WNDCLASSW wc = { 0 };

wc.hbrBackground = (HBRUSH)COLOR\_WINDOW;

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

wc.hInstance = hInst;

wc.lpszClassName = L"MainWndClass";

wc.lpfnWndProc = MainWindowProc;

if (!RegisterClassW(&wc))

return -1;

CreateWindowW(

L"MainWndClass", L"Блокнот",

WS\_OVERLAPPEDWINDOW | WS\_VISIBLE,

400, 200, 500, 250,

NULL, NULL, NULL, NULL

);

MSG msg = { 0 };

while (GetMessage(&msg, NULL, NULL, NULL)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

void Editor(HWND hWnd)

{

hEdit = CreateWindowW(L"Edit", L"", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | WS\_BORDER | WS\_VSCROLL | WS\_HSCROLL,

10, 50, 400, 300, hWnd, NULL, NULL, NULL);

}

void Menubar(HWND hWnd) {

HMENU MainMenu = CreateMenu();

HMENU FileMenu = CreateMenu();

HMENU OpenMenu = CreateMenu();

HMENU EditMenu = CreateMenu();

AppendMenuW(MainMenu, MF\_POPUP, (UINT\_PTR)FileMenu, L"File");

AppendMenuW(FileMenu, MF\_POPUP, (UINT\_PTR)OpenMenu, L"Open");

AppendMenuW(OpenMenu, MF\_STRING, MB\_TXT, L".txt");

AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenuW(FileMenu, MF\_STRING, MB\_SAVE, L"Save");

AppendMenuW(FileMenu, MF\_STRING, MB\_SAVEAS, L"Save as");

AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenuW(FileMenu, MF\_STRING, MB\_CLOSE, L"Close");

AppendMenuW(FileMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenuW(FileMenu, MF\_STRING, MB\_EXIT, L"Exit");

AppendMenuW(MainMenu, MF\_POPUP, (UINT\_PTR)EditMenu, L"Edit");

AppendMenuW(EditMenu, MF\_STRING, MB\_CUT, L"Cut");

AppendMenuW(EditMenu, MF\_STRING, MB\_COPY, L"Copy");

AppendMenuW(EditMenu, MF\_STRING, MB\_PASTE, L"Paste");

AppendMenuW(EditMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenuW(EditMenu, MF\_STRING, MB\_SELALL, L"Select all");

SetMenu(hWnd, MainMenu);

}

void DisplayTXT(wchar\_t\* path)

{

FILE\* file = \_wfopen(path, L"rt");

if (file == NULL)

{

return;

}

fseek(file, 0, SEEK\_END);

int \_size = ftell(file) / sizeof(wchar\_t);

rewind(file);

wchar\_t\* data = new wchar\_t[\_size + 1];

fread(data, sizeof(wchar\_t), \_size, file);

data[\_size] = '\0';

SetWindowTextW(hEdit, data);

delete[] data;

fclose(file);

}

void OpenTXT(HWND hWnd)

{

OPENFILENAMEW ofn;

wchar\_t file\_name[100];

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = file\_name;

ofn.lpstrFile[0] = L'\0';

ofn.nMaxFile = MAX\_PATH;

ofn.lpstrFilter = L"txt Files\00\*.TXT\0";

ofn.nFilterIndex = 1;

GetOpenFileNameW(&ofn);

DisplayTXT(ofn.lpstrFile);

}

void WriteTXT(wchar\_t\* path)

{

FILE\* file = \_wfopen(path, L"w+t");

int \_size = GetWindowTextLengthW(hEdit);

LPWSTR data = new WCHAR[\_size+1];

GetWindowTextW(hEdit, data, \_size+1);

data[\_size] = L'\0';

fwrite(data, sizeof(wchar\_t), \_size, file);

delete[] data;

time\_t t;

time(&t);

wchar\_t \*str = \_wctime(&t);

fwrite(str, sizeof(wchar\_t), wcslen(str), file);

delete[] str;

fclose(file);

}

void SavetoTXT(HWND hWnd)

{

OPENFILENAMEW ofn;

wchar\_t file\_name[100];

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = file\_name;

ofn.lpstrFile[0] = L'\0';

ofn.nMaxFile = MAX\_PATH;

ofn.lpstrFilter = L"txt Files\00\*.TXT\0";

ofn.nFilterIndex = 1;

GetSaveFileNameW(&ofn);

WriteTXT(ofn.lpstrFile);

}

LRESULT CALLBACK MainWindowProc(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {

RECT rct;

switch (msg)

{

case WM\_CREATE: {

Menubar(hWnd);

Editor(hWnd);

break;

}

case WM\_COMMAND: {

switch (wp) {

case MB\_TXT: {

OpenTXT(hWnd);

break;

}

case MB\_SAVE: {

SavetoTXT(hWnd);

break;

}

case MB\_CUT: {

SendMessage(hEdit, WM\_CUT, 0, 0);

break;

}

case MB\_PASTE: {

SendMessage(hEdit, WM\_PASTE, 0, 0);

break;

}

case MB\_COPY: {

SendMessage(hEdit, WM\_COPY, 0, 0);

break;

}

case MB\_SELALL: {

SendMessage(hEdit, EM\_SETSEL, 0, -1);

break;

}

case MB\_EXIT: {

PostQuitMessage(0);

break;

}

}

break;

}

case WM\_CLOSE: {

PostQuitMessage(0);

break;

}

case WM\_SIZE: {

MoveWindow(hEdit, 0, 0, LOWORD(lp), HIWORD(lp), FALSE);

break;

}

}

return DefWindowProcW(hWnd, msg, wp, lp);

}

Header.h

#pragma once

#define MB\_TXT 100

#define MB\_SAVE 101

#define MB\_SAVEAS 102

#define MB\_CLOSE 103

#define MB\_EXIT 104

#define MB\_CUT 112

#define MB\_COPY 113

#define MB\_PASTE 114

#define MB\_SELALL 115

#define IDC\_OPEN 10

#define IDC\_SAVE 11

#define IDC\_SAVEAS 12

#define IDC\_EXIT 13

#define IDC\_ABOUT 14

#define ID\_EDIT 20

#define ID\_STATUS 21