******

Колледж космического машиностроения и технологии

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка компьютерной программы "Текстовый редактор»

Выполнил студент

Щепкин Михаил Викторович

Группа П2-18

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

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Проверил преподаватель:

Гусятинер Леонид Борисович

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

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

Королев, 2021

**Приложение к заданию на курсовой проект**

**Краткое описание программы:**

Приложение блокнот на ПК под управлением OC windows, написанное с использованием языка Python и библиотек Tkinter и PIL

**Разработать для программы:**

1. Главное окно с интерфейсом, содержащее в себе подменю: правку, файл, шрифт, помощь и кнопку выход
2. Работу с текстовыми документами: сохранение файлов, открытие файлов и создание нового файла
3. Кнопки отмены действия, возврата действия, полужирного и курсивного шрифта
4. Чтение файлов формата, txt, py и html

**Оглавление**

[Введение 5](#_Toc75567258)

[1. Теоретическая часть 6](#_Toc75567259)

[1.1 Определение требований к программному средству 6](#_Toc75567260)

[1.2 Спецификация качества программного обеспечения 6](#_Toc75567261)

[1.3 Функциональная спецификация 7](#_Toc75567262)

[2. Проектная часть 8](#_Toc75567263)

[2.1 Выбор инструментария 8](#_Toc75567264)

[2.2 Язык программирования Python 8](#_Toc75567265)

[2.3 Преимущества и недостатки языка Python 9](#_Toc75567266)

[2.4 Библиотеки Tkiner и PIL 11](#_Toc75567267)

[2.5 Основные требования к проекту 12](#_Toc75567268)

[2.5.1 Требования к эргономике и технической эстетики 12](#_Toc75567269)

[2.5.2 Стадии и этапы разработки 12](#_Toc75567270)

[3. Проектирование 14](#_Toc75567271)

[3.1 Проектирование архитектуры приложения 14](#_Toc75567272)

[3.2 Диаграмма классов 15](#_Toc75567273)

[3.3 Диаграмма прецедентов 16](#_Toc75567274)

[3.4 Проектирование сценария 17](#_Toc75567275)

[3.5 Проектирование пользовательского интерфейса 18](#_Toc75567276)

[4.1 Спецификация классов 26](#_Toc75567277)

[4.2 Описание алгоритма на естественном языке 27](#_Toc75567278)

[5. Тестирование 28](#_Toc75567279)

[5.1 Разработка тестов 28](#_Toc75567280)

[6. Руководство пользователя 29](#_Toc75567281)

[6.1 Назначение программного средства 29](#_Toc75567282)

[6.2 Порядок установки программного средства 29](#_Toc75567283)

[6.3 Руководство пользователя 29](#_Toc75567284)

[Заключение 30](#_Toc75567285)

[Список литературы и источников 31](#_Toc75567286)

[Приложение 1 32](#_Toc75567287)

[Приложение 2 34](#_Toc75567288)

[Приложение 3 35](#_Toc75567289)

[Приложение 4 37](#_Toc75567290)

[Приложение 5 38](#_Toc75567291)

[Приложение 6 40](#_Toc75567292)

# Введение

В данной пояснительной записке рассматривается описание программы   
«Текстовый редактор». Программа очень проста в использовании и имеет простой и интуитивно - понятный интерфейс. В качестве основного инструмента разработки применяется sublime text и idle. Язык программирования Python.  
первой части будут рассмотрены:

Во второй части будут рассмотрены инструменты и модули, которые были разработаны, структура программной части и листинги ключевых частей программных модулей.

В третьей части будет освещено взаимодействие пользователя с программой, приведены изображения графического интерфейса программы и сообщения оператору. В заключительной части будут сделаны общие выводы

# 1. Теоретическая часть

## 1.1 Определение требований к программному средству

программный обеспечение архитектура руководство

Программа должна создавать и открывать текстовые документы, форматировать и редактировать текст. Также должна быть предусмотрена возможность сохранения текста в файл.

## 1.2 Спецификация качества программного обеспечения

Функциональность: программа должна являться полностью завершенным программным продуктом, не требующим каких-либо дополнительных специальных компонентов. Надёжность: программа должна быть автономной.

Эффективность: программа должна предъявлять минимальные требования к аппаратному обеспечению, требования к аппаратным ресурсам должны быть намного ниже, чем для самой ОС. Программа должна иметь интуитивно понятный пользовательский графический интерфейс.

## 1.3 Функциональная спецификация

Минимальные технические характеристики компьютера, на котором гарантируется стабильная работа программы:

* Компьютер с процессором класса Pentium II 450 МГц;
* Память: 64 МБ ОЗУ;
* Монитор Super VGA с разрешением 800x800 точек или более высоким, поддерживающий 256 цветов;
* Операционная система: операционная система Windows XP Service Pack 3, необходимо наличие Microsoft .NET Framework Runtime v4.0;
* Наличие свободного дискового пространства на жёстком.

Внешние функции:

* открытие и сохранение текстовых документов;
* форматирование и редактирование текста;
* программа должна выполнять все функции за конечное количество времени.

# 2. Проектная часть

## 2.1 Выбор инструментария

Углубление знаний в Python. Создание текстового редактора, работа с текстом, работа с библиотеками Tkinter и PIL

## 2.2 Язык программирования Python

Python – это высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным — всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Автором языка Python является Гвидо Ван Россум, язык был разработан 20 февраля в 1991 году, Гвидо Ван Россумом вместе с Python Software Foundation.

## 2.3 Преимущества и недостатки языка Python

К многочисленным преимуществам языка Python относят:

* Язык характеризуется логичным синтаксисом, вследствие чего исходный код программ, написанных «на питоне», легко читается и воспринимается.
* Еще одно из преимуществ данного языка программирования — его условная легкость. Он считается наиболее подходящим для начинающих специалистов: разрабатывать несложные программы можно научиться уже спустя пару-тройку дней изучения.
* Большое интернет-сообщество. Если разработчик сталкивается с вопросами и трудностями, он всегда может спросить совета у коллег, что значительно ускоряет решение проблем.
* Гибкость и масштабируемость. Python позволяет разработчикам адаптировать высокоуровневую логику приложения, что позволяет легко расширять сложные приложения по мере необходимости.
* Разработка «на питоне» идет быстрее, чем на большинстве других языках.
* Python является интерпретируемым языком программирования. Это значит, что до запуска он представляет собой обычный текстовый файл. Соответственно, программировать можно почти на всех платформах.

К недостаткам языка Python можно отнести:

* Скорость работы. Высокопроизводительные проекты на чистом Python написать будет затруднительно. Для этого требуется прибегать к помощи других языков.
* Безопасность, которая обеспечивается моделью памяти языка Python, сводит на нет большинство возможных процессорных оптимизаций.

Это 2 главных минуса языка программирования Python: его малое быстродействие и недостаточные возможности статического анализа кода. Эти недостатки являются связанными между собой, и решение последней проблемы автоматически откроет двери для устранения первой.

## 2.4 Библиотеки Tkiner и PIL

Tkinter  – это кроссплатформенная библиотека для разработки графического интерфейса на языке Python. Tkinter расшифровывается как Tk interface, и является интерфейсом к [Tcl/Tk](http://ru.wikipedia.org/wiki/Tcl).  
Tkinter входит в стандартный дистрибутив Python.

PIL (Python Image Library) является сторонней библиотекой обработки изображений для Python, но благодаря своим мощным функциям и многочисленным пользователям она почти считается официальной библиотекой обработки изображений Python

## 2.5 Основные требования к проекту

Программа должна создавать и открывать текстовые документы, форматировать и редактировать текст. Также должна быть предусмотрена возможность сохранения текста в файл.

### 2.5.1 Требования к эргономике и технической эстетики

Программа легко осваиваемая и проста в эксплуатации.

### 2.5.2 Стадии и этапы разработки

Разработка технического задания.

Проектирование.

Разработка классов.

Дизайн главного меню и интерфейса.

Написание кода, исправление ошибок, корректировка дизайна.

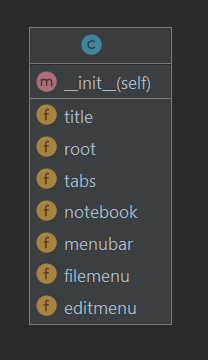


Рисунок 1 – «Диаграмма классов»

# 3. Проектирование

## 3.1 Проектирование архитектуры приложения

Данное программное средство можно разбить на подсистемы, взаимодействующие друг с другом в соответствии с (Рисунком 1);



Рисунок 1 - «Структура программы»

Подсистема управления текстом представляет собой операции над текстом. Возможны операции: добавление, удаление, редактирование, форматирование. Подсистема интерфейса реализует пользовательский интерфейс программы.

## 3.2 Диаграмма классов

В данном разделе располагается диаграмма класса текстового редактора. (Рисунок 2);

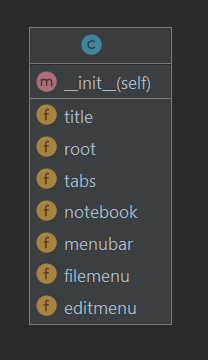


Рисунок 2 – «Диаграмма классов»

## 3.3 Диаграмма прецедентов

В данном разделе содержится диаграмма прецедентов для приложения, на которой показаны возможные функциональные отношения. (Рисунок 3);

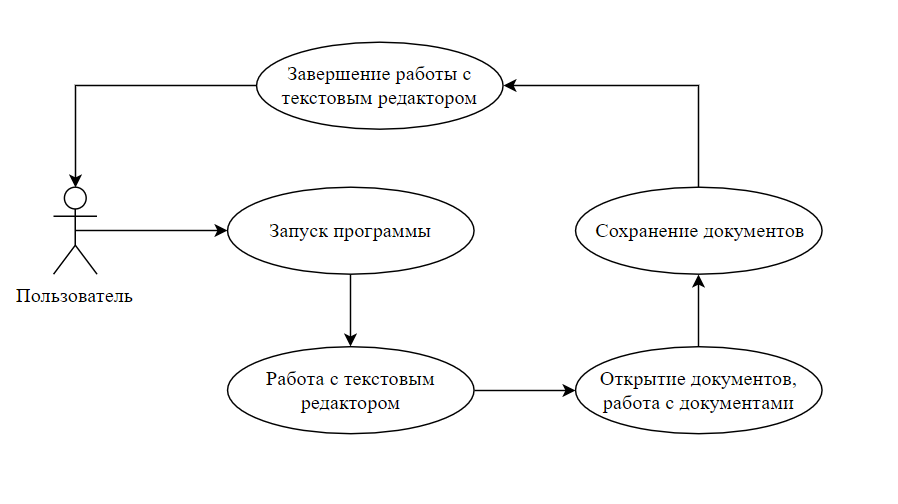


Рисунок 3 - «Диаграмма прецедентов текстового редактора»

## 3.5 Блок – схема для функции сохранения файла, «SaveFile»

Листинг 1.

#Функция сохранить файл

def SaveFile(event = 's'):

global OpenStatusName

if OpenStatusName:

textFile = open(OpenStatusName, 'w')

textFile.write(text.get(1.0, END))

statusBar.config(text = f'Saved: {OpenStatusName} ')

showinfo(title = "TextEditor", message = "Ваш файл успешно сохранён" )

else:

SaveAs()

textFile.close()

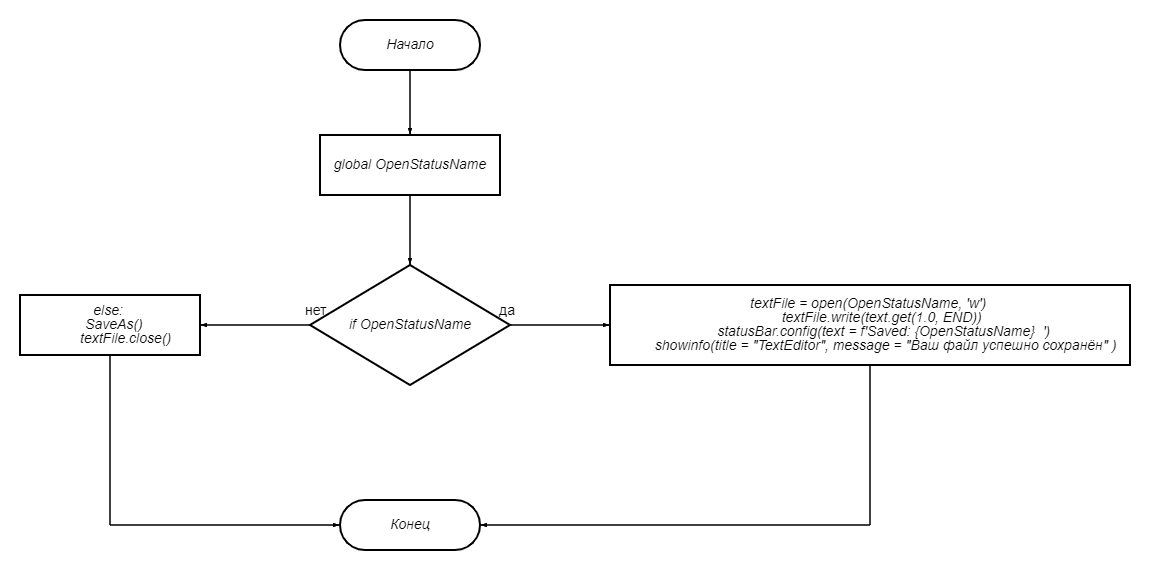


Рисунок 4 - Блок схема функции «SaveFile»

## 3.4 Проектирование сценария

В данном разделе приведен сценарий использования текстового редактора пользователем (Рисунок 5).

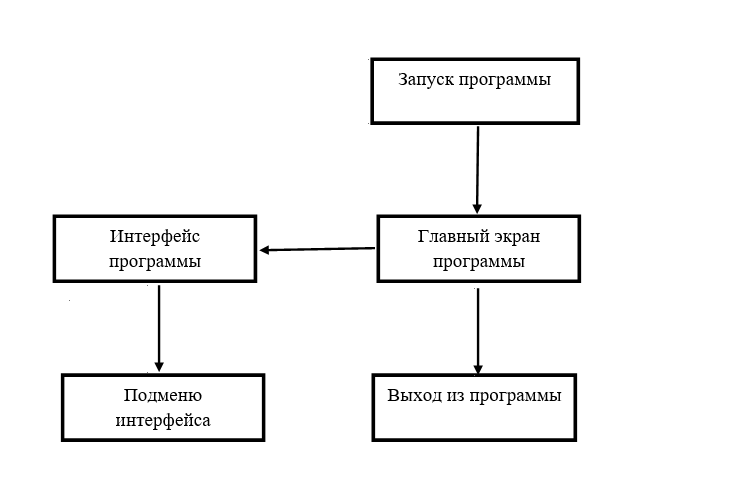


Рисунок 5 – «Сценарий использования блокнота»

## 3.5 Проектирование пользовательского интерфейса

Главная форма содержит основные элементы интерфейса и даёт доступ к большинству функциям. (Рисунок 6);

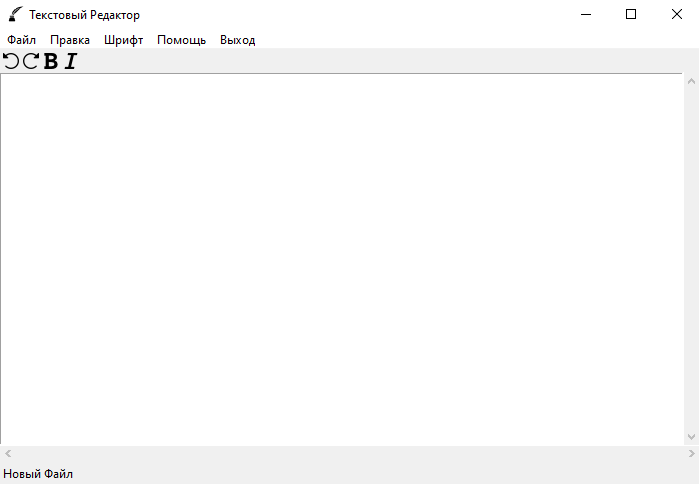


Рисунок 6 - «Главная форма программы»

Форма содержит панель с ниспадающими списками, при помощи которых осуществляется доступ к основным функциям программы.

Главное меню содержит следующие вкладки:

«Файл»: подменю включает в себя кнопки выполнения создания, открытия, сохранения файла и сохранение файла как. (Рисунок 7);

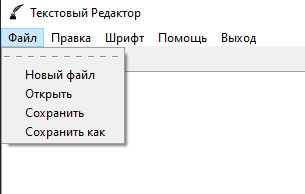


Рисунок 7 - Подменю «Файл»

«Правка»: подменю включает в себя возможность отменить последнее действие, а так же вырезать, скопировать и вставить текст. (Рисунок 8);

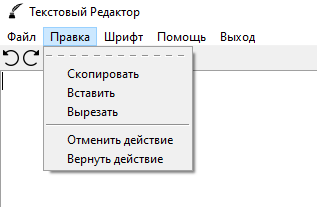


Рисунок 8 - Подменю «Правка»

«Шрифт»: подменю включает в себя возможность включения / выключения режима «Полужирный шрифт» а так же «Курсивный шрифт» (Рисунок 9);

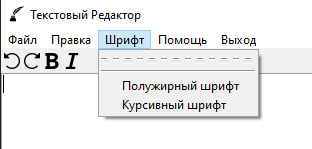


Рисунок 9 - Подменю «Шрифт»

«Помощь»: подменю включает в себя комбинации клавиш которые можно использовать для, сохранение, открытия, сохранения как, создание нового файла, копирование, вставка, вырезать и т.д (Рисунок 10);

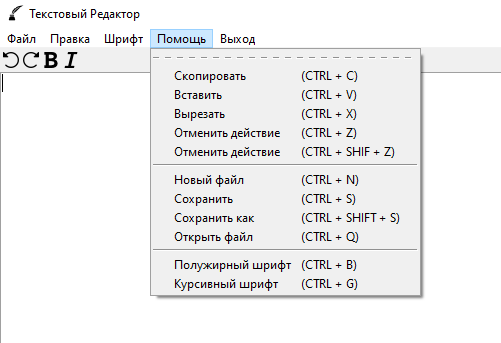


Рисунок 10 - Подменю «Помощь»

«Выход»: кнопка позволяющая выйти из программы сразу же. (Рисунок 11);

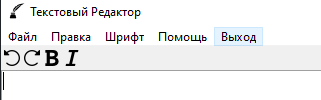


Рисунок 11 - Кнопка «Выход»

Под подменю ещё находится интерфейс содержащий 4 кнопки: (Рисунок 12);

* Отменить действие;
* Вернуть действие;
* Полужирный шрифт;
* Курсивный шрифт

****

Рисунок 12 - Кнопки «Отмены, возврата действий, полужирный и курсивный шрифты»

Ещё в главном интерфейсе блокнота содержится путь открытого файла:

(Рисунок 13);



Рисунок 13 - «Путь к файлу»**4. Реализация**

## 4.1 Спецификация классов

Подсистема интерфейса - объединяет все остальные подсистемы. Здесь реализован весь пользовательский интерфейс. Основные методы, реализованы в

## 4.2 Описание алгоритма на естественном языке

Сразу после загрузки Текстовый редактор автоматически создает пустой текстовый документ, имеющий по умолчанию название Новый Файл. Создание текстового документа происходит с помощью отчистки области набора текста.

# 5. Тестирование

## 5.1 Разработка тестов

Для тестирования программы искусственно создавались критическая ситуации. Цель тестирования: убедится что программа обрабатывает все критические ситуации и не завершает работу.

# 6. Руководство пользователя

## 6.1 Назначение программного средства

Данное программное средство позволяет решать прикладные задачи по работе с текстовыми документами.

## 6.2 Порядок установки программного средства

Программное средство не требует сложной процедуры установки. Установка происходит путем копирования файлов программы в любой раздел жесткого диска компьютера. Важным условием является сохранность целостности компонентов программы.

## 6.3 Руководство пользователя

Программа проста в использовании и имеет достаточно необходимых функций. Основное пространство рабочего окна редактора занимает область набора текста, в которой отображается мигающий курсор, сразу после загрузки

Текстовый редактор автоматически создает пустой текстовый документ, имеющий по умолчанию название Новый Файл. Вы можете открыть для редактирования уже существующий текстовый файл, воспользовавшись пунктом Открыть в меню Файл. Все функции программы можно вызвать посредством ниспадающего меню в верхней части программы, а так же горячими клавишами.

# Заключение

В рамках курсовой работы была разработана программа для создания и редактирования текстовых документов. В процессе проектирования использовались методологии модульного и объектно-ориентированного программирования.

Для разработки данной программы нужно было грамотно спроектировать внутреннюю архитектуру программы и реализацию необходимых функций и классов.

При разработке данного программного средства применялся язык программирования Python как один из наиболее современных и перспективных языков программирования.

# Список литературы и источников

<https://stackoverflow.com/questions/9561030/vertical-and-horizontal-scrollbars-on-tkinter-widget>

<https://www.flaticon.com/search?word=italic&type=icon>

<https://pythononline.ru/osnovy/button-tkinter>

<https://www.youtube.com/watch?v=XW65JTd8UgI&t=253s>

<https://www.youtube.com/watch?v=721wxwOOdw8&t=273s>

<https://www.youtube.com/watch?v=w5Nd4O76tDw&t=358s>

<https://www.youtube.com/watch?v=rUgAC_Ssflw>

<https://pythonru.com/uroki/sozdanie-maketov-okna-tkinter-5#Geometry-manager-Pack>

# Приложение 1

Листинг функций сохранения и сохранения как

#Функция сохранить файл

def SaveFile(event = 's'):

global OpenStatusName

if OpenStatusName:

textFile = open(OpenStatusName, 'w')

textFile.write(text.get(1.0, END))

statusBar.config(text = f'Saved: {OpenStatusName} ')

showinfo(title = "TextEditor", message = "Ваш файл успешно сохранён" )

else:

SaveAs()

textFile.close()

#Функция сохранить файл как

def SaveAs(event = 's'):

textFile = filedialog.asksaveasfilename(defaultextension = ".\*", title = "Сохранить файл как", filetypes = (("Text Files", "\*.txt"),("HTML Files","\*.html"),("Python Files", "\*.py"),("All Files", "\*.\*")))

if textFile:

name = textFile

statusBar.config(text = f'Saved: {name} ')

showinfo(title = "TextEditor", message = "Ваш файл успешно сохранён" )

name = name.replace("C:/", "")

root.title(f' {name} ')

textFile = open(textFile, 'w')

textFile.write(text.get(1.0, END))

textFile.close()

# Приложение 2

Функции создания нового файла и открытия файла

#Функция новый файл

def NewFile(event = 'n'):

text.delete('1.0', END)

root.title('Новый Файл')

statusBar.config(text = "Новый Файл - TextEditor")

global OpenStatusName

OpenStatusName = False

#Функция открыть файл

def OpenFile(event = 'q'):

text.delete('1.0', END)

textFile = filedialog.askopenfilename(title = "Открыть Файл", filetypes = (("Text Files", "\*.txt"),("HTML Files","\*.html"),("Python Files", "\*.py"),("All Files", "\*.\*")))

if textFile:

global OpenStatusName

OpenStatusName = textFile

name = textFile

statusBar.config(text = f'{name} ')

root.title('TextEditor')

textFile = open(textFile, 'r')

stuff = textFile.read()

text.insert(END, stuff)

textFile.close()

# Приложение 3

Функции копировать, вырезать, вставить, отмены действия и возврата действия

#Функция копировать

def copy(event = None):

global selected

if event:

selected = root.clipboard\_get()

if text.selection\_get():

selected = text.selection\_get()

root.clipboard\_append(selected)

#Функция вырезать

def cut(event = None):

global selected

if event:

selected = root.clipboard\_get()

else:

if text.selection\_get():

selected = text.selection\_get()

text.delete("sel.first", "sel.last")

root.clipboard\_clear()

root.clipboard\_append(selected)

#Функция вставить

def paste(event = None):

global selected

if event:

selected = root.clipboard\_get()

else:

if selected:

position = text.index(INSERT)

text.insert(position, selected)

#Функция отменить действие

def undo(event = None):

text.edit\_undo

#Функция возврата действия

def redo(event = None):

text.edit\_redo

# Приложение 4

Функции полужирного и курсивного шрифта

#Функция полужирного шрифта

def BoldFont(event = "b"):

bold = font.Font(text, text.cget("font"))

bold.configure(weight = "bold")

text.tag\_configure("bold", font = bold)

current\_tags = text.tag\_names("sel.first")

if "bold" in current\_tags:

text.tag\_remove("bold", "sel.first", "sel.last")

else:

text.tag\_add("bold", "sel.first", "sel.last")

#Функция курсивного шрифта

def ItalicsFont(event = "g"):

italics = font.Font(text, text.cget("font"))

italics.configure(slant = "italic")

text.tag\_configure("italic", font = italics)

current\_tags = text.tag\_names("sel.first")

if "italic" in current\_tags:

text.tag\_remove("italic", "sel.first", "sel.last")

else:

text.tag\_add("italic", "sel.first", "sel.last")

# Приложение 5

Кнопки отмены действия, возврата действия, полужирного и курсивного шрифта

#Отмена действия

#undoImage = ImageTk.PhotoImage(file = "C:/Users/p2-18/Desktop/TextEditor/undo.png")

undoImage = ImageTk.PhotoImage(file = "C:/TextEditor/undo.png")

undoButton = Button(root, image = undoImage, command = text.edit\_undo, relief = 'flat')

undoButton.pack(anchor = W)

#Возврат действия

#redoImage = ImageTk.PhotoImage(file = "C:/Users/p2-18/Desktop/TextEditor/redo.png")

redoImage = ImageTk.PhotoImage(file = "C:/TextEditor/redo.png")

redoButton = Button(root, image = redoImage, command = text.edit\_redo, relief = 'flat')

redoButton.place(x = 20)

#Кнопка шрифт полужирный

#boldImage = ImageTk.PhotoImage(file = "C:/Users/p2-18/Desktop/TextEditor/bold.png")

boldImage = ImageTk.PhotoImage(file = "C:/TextEditor/bold.png")

boldButton = Button(root, image = boldImage, command = BoldFont, relief = 'flat')

boldButton.place(x = 40)

#Кнопка шрифт курсивный

#italicsImage = ImageTk.PhotoImage(file = "C:/Users/p2-18/Desktop/TextEditor/italic.png")

italicsImage = ImageTk.PhotoImage(file = "C:/TextEditor/italic.png")

italicsButton = Button(root, image = italicsImage, command = ItalicsFont, relief = 'flat')

italicsButton.place(x = 60)

# Приложение 6

Бинды используемые в подменю файла, правки, шрифта

root.bind('<Control - Key - n>', NewFile)

root.bind('<Control - Key - s>', SaveFile)

root.bind('<Control - Key - q>', OpenFile)

root.bind('<Control - Key - S>', SaveAs)

root.bind('<Control - Key - c>', copy)

root.bind('<Control - Key - v>', paste)

root.bind('<Control - Key - x>', cut)

root.bind('<Control - Key - z>', undo)

root.bind('<Control - Key - Z>', redo)

root.bind('<Control - Key - b>', BoldFont)

root.bind('<Control - Key - g>', ItalicsFont)