МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіотехнічних систем літальних апаратів

Кафедра комп’ютерних систем та мереж

**Курсова робота**

|  |  |
| --- | --- |
| з | *Об’єктно–орієнтованого програмування* |
|  | (назва дисципліни) |
| на тему | *«Формування реферату пояснювальної записки*» |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Виконав: студент | | | *2* | курсу групи № | | | | *525a* |
| напряму підготовки (спеціальності) | | | | | | | | |
| *150102 – комп’ютерна інженерія* | | | | | | | | |
| (шифр і назва напряму підготовки (спеціальності)) | | | | | | | | |
| *Голубов Р.* | | | | | | | | |
| (прізвище й ініціали студента) | | | | | | | | |
| Керівник: | *старший викладач* | | | | | | | |
| *Бабешко Євген Васильович* | | | | | | | | |
| (посада, науковий ступінь, прізвище й ініціали) | | | | | | | | |
| Національна шкала: | | | | |  | | | |
| Кількість балів: | | | | |  | | | |
| Оцінка ECTS: | | | | |  | | | |
| Члени комісії: | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |
|  | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |
|  | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |

Харків – 2017

Міністерство освіти і науки України

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

|  |  |  |
| --- | --- | --- |
| Факультет | *радіотехнічних систем літальних апаратів* | |
|  | (повне найменування) | |
| Кафедра | *комп’ютерних систем та мереж* | |
|  | (повне найменування) | |
| Рівень вищої освіти | | *перший (бакалаврський)* |
|  | | (назва) |
| Напрям підготовки | | *150102 – комп’ютерна інженерія* |
|  | | (код та найменування) |

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ**

|  |  |  |
| --- | --- | --- |
| *Голубов Рамiр* | | |
| (прізвище, ім’я, по батькові) | | |
| 1. Тема проекту (роботи) | *«Генератор документiв про вищу освiту»* | |
|  | | |
|  | | |
| Керівник проекту (роботи) | | *Бабешко Євген Васильович старший викладач* |
|  | | ( прізвище, ім’я, по батькові, науковий ступінь, вчене звання) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2. Строк подання студентом проекту (роботи) | | | *25.05.2017* | |
| 3. Вихідні дані до проекту (роботи) | |  | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
| 4. Зміст пояснювальної записки (перелік завдань, які потрібно розв’язати) | | | |  |
| *Реферат, Постановка задачі, Проектування, Розробка, Верифікація та* | | | | |
| *тестування, Висновок, Перелік посилань, Додаток А. Технічне завдання,* | | | | |
| *Додаток Б. Вихідні тексти програми, Додаток В. Керівництво Оператора,* | | | | |
| *Додаток Г. Презентація* | | | | |
|  | | | | |
|  | | | | |
| 5. Перелік графічного матеріалу | *діаграма варіантів використання,* | | | |
| *діаграма послідовності, діаграма пакетів, діаграма класів* | | | | |
|  | | | | |
| *.* | | | | |
|  | | | | |

6. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
| *Основний* | *Бабешко Є. В. старший викладач* | *24.01.2017* | *24.01.2017* |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Нормоконтроль |  | *Бабешко Є. В.* | *«* | *25* | *»* | *05* | *20* | *17* | *р* |
|  | (підпис) | (ініціали та прізвище) |  |  |  |  |  |  |  |

7. Дата видачі завдання « 24 » 01 2017 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів курсового  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| *1* | *Реферат* | *24.01.2017* |  |
| *2* | *Постанова задачі* | *07.02.2017* |  |
| *3* | *Розробка ТЗ* | *21.02.2017* |  |
| *4* | *Проектування* | *07.03.2017* |  |
| *5* | *Розробка* | *21.03.2017* |  |
| *6* | *Тестування та верифікація* | *04.04.2017* |  |
| *7* | *Висновки* | *18.04.2017* |  |
| *8* | *Розробка керівництва оператора* | *02.04.2017* |  |
| *9* | *Оформлення пояснювальної записки* | *16.04.2017* |  |
| *10* | *Розробка презентації* | *16.04.2017* |  |

|  |  |  |
| --- | --- | --- |
| **Студент** |  | *Голубов Р.* |
|  | (підпис) | (прізвище та ініціали) |
| **Керівник проекту (роботи)** |  | *Бабешко Є.В.* |
|  | (підпис) | (прізвище та ініціали) |

РЕФЕРАТ

Пояснительная записка на курсовой проект: 47 с., 31 ил., 19 табл., 4 приложения, 5 источников.

Данная работа посвящена задаче по созданию генератора реферата объяснительной записки.

Цель работы *–* разработка приложения, которое может быть использовано для формирования реферата пояснительной записки.

Приложение было реализовано с помощью языка программирования С#. За графический интерфейс отвечал Windows Forms. Для доступа к объектам файла Word использовалась библиотека Microsoft.Interop.Word.dll.

Ключевые слова: MS WORD, OFFICE.

Содержание

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc484445558)

[1.1 Анализ предметной области 5](#_Toc484445559)

[1.2 Анализ существующих решений 8](#_Toc484445560)

[1.3 Постановка задачи 8](#_Toc484445561)

[2 ПРОЕКТИРОВАНИЕ 9](#_Toc484445562)

[2.1 Выбор и обоснование архитектуры 9](#_Toc484445563)

[2.2 Диаграмма прецедентов 11](#_Toc484445564)

[2.3 Диаграмма последовательности 12](#_Toc484445565)

[2.4 Проектирование графического интерфейса 13](#_Toc484445566)

[3 РАЗРАБОТКА 16](#_Toc484445567)

[3.1 Диаграмма пакетов 16](#_Toc484445568)

[3.2 Диаграмма классов 17](#_Toc484445569)

[3.3 Разработка и описание ключевых методов классов 18](#_Toc484445570)

[4 ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММЫ 22](#_Toc484445571)

[4.1 Верификация 22](#_Toc484445572)

[4.2 Тестирование программы 25](#_Toc484445573)

[ЗАКЛЮЧЕНИЕ 25](#_Toc484445574)

[ПЕРЕЧЕНЬ ССЫЛОК 27](#_Toc484445575)

[ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 28](#_Toc484445576)

[ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ КОДЫ ПРОГРАММЫ 31](#_Toc484445577)

[ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 37](#_Toc484445578)

[ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ 38](#_Toc484445579)

# ПОСТАНОВКА ЗАДАЧИ

## 1.1 Анализ предметной области

Текстовый файл — компьютерный [файл](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), содержащий [текстовые данные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), как правило, организованные в виде строк. Текстовым файлам противопоставляются [бинарные файлы](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB), которые могут содержать любой тип данных, закодированных в двухчастной форме в компьютерных целях хранения и обработки. Много форматов бинарного файла содержат части, которые могут интерпретироваться как текст; например, некоторые компьютерные файлы документа, содержащие форматированный текст, такие как более старые файлы документа Microsoft Word, содержат текст документа, но также и содержат информацию о форматировании в двухчастной форме.

В отличие от термина «[текстовые данные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5)» («текстовый формат данных»), характеризующего содержимое данных, термин «текстовый файл» относится к файлу и характеризует его как контейнер, хранящий такие данные.

Текстовый файл содержит последовательность символов (в основном печатных знаков, принадлежащих тому или иному набору символов). Эти символы обычно сгруппированы в строки. В современных системах строки разделяются разделителями строк, в прошлом же применялось хранение строк в виде записей постоянной или переменной длины. Иногда конец текстового файла (особенно если в файловой системе не хранится информация о размере файла) также отмечается одним или более специальными знаками, известными как маркеры конца файла. [1].

Microsoft Word (часто — MS Word, WinWord или просто Word) — [текстовый процессор](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80), предназначенный для создания, просмотра и редактирования [текстовых документов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB), с локальным применением простейших форм [таблично](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0)–[матричных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) алгоритмов. Выпускается [корпорацией Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в составе [пакета](https://ru.wikipedia.org/wiki/%D0%9E%D1%84%D0%B8%D1%81%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82) [Microsoft Office](https://ru.wikipedia.org/wiki/Microsoft_Office" \o "Microsoft Office). Первая версия была написана [Ричардом Броди](https://ru.wikipedia.org/w/index.php?title=%D0%91%D1%80%D0%BE%D0%B4%D0%B8,_%D0%A0%D0%B8%D1%87%D0%B0%D1%80%D0%B4&action=edit&redlink=1) ([Richard Brodie](https://en.wikipedia.org/wiki/Richard_Brodie" \o "en:Richard Brodie)) для [IBM PC](https://ru.wikipedia.org/wiki/IBM_PC), использующих [DOS](https://ru.wikipedia.org/wiki/DOS), в 1983 году. Позднее выпускались версии для [Apple Macintosh](https://ru.wikipedia.org/wiki/Macintosh" \o "Macintosh)(1984), [SCO UNIX](https://ru.wikipedia.org/wiki/SCO_UNIX) и [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows) (1989). Текущей версией является Microsoft Office Word 2016 для [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows) и [macOS](https://ru.wikipedia.org/wiki/MacOS" \o "MacOS).

Microsoft Word является наиболее популярным из используемых в данный момент текстовых процессоров, что сделало его бинарный формат документа стандартом де–факто, и многие конкурирующие программы имеют поддержку совместимости с данным форматом. Расширение «.doc» на платформе IBM PC стало синонимом двоичного формата Word 97—2000. .doc или .DOC (от англ. document) — расширение имени файла, используемое для файлов, представляющих текст, с разметкой или без. Расширение «.doc» часто использовалось для обозначения текстовых файлов, содержащих текст без форматирования, позже стало использоваться для двоичных форматов с разметкой. Фильтры экспорта и импорта в данный формат присутствуют в большинстве текстовых процессоров. Формат документа разных версий Word меняется, различия бывают довольно тонкими [2].

Формат Rich Text Format (RTF) позволяет пользователям набирать текст, а также сохранять и копировать его в другие операционные системы. В отличие от других текстовых форматов формат RTF может содержать только текстовые данные. При этом текст можно набрать в любых других форматах, а после сохранить в формате RTF. Формат позволяет легко переносить текст из одних программ в другие (например, Microsoft Office или Word Perfect), а также из одних версий ОС Windows в другие. При этом не стоит путать формат RTF с такими форматами как enriched text, Rich text или RFT–DCA от компании IBM.

Первая версия формата RTF 1.1 была выпущена в 1987 году. В марте 2008 г. была выпущена версия RTF 1.9.1. Стоит отметить, что почти каждая новая версия Microsoft Office Word обновляет версию данного формата. Кроме того, почти в каждой новой версии добавляется новая группа контрольных слов. В самой последней версии внедрена возможность использования XML, XLS, OMML, а также элементов SmartTags. Кроме того, пользователь может внедрять в файл данные о шрифтах, символы Unicode (16 бит), а также изображения (на бинарном уровне). При просмотре файла в режиме просмотра обычного текста язык разметки виден в гораздо меньшей степени и не отвлекает читателя от основного текста.

При разработке решений Word в Visual Studio выполняется взаимодействие с объектной моделью Word. Эта объектная модель состоит из классов и интерфейсов, которые предоставляются в основной сборке взаимодействия для Word и определены в пространстве имен Microsoft.Office.Interop.Word.

Word предоставляет сотни различных объектов, с которыми можно взаимодействовать. Эти объекты организованы в виде иерархии, которая точно соответствует пользовательскому интерфейсу. В верхней части иерархии находится объект [Application](https://msdn.microsoft.com/ru-ru/library/microsoft.office.interop.word.application.aspx). Этот объект представляет текущий экземпляр Word. Ядро автоматизации UI маскирует любые отличия в структурах, принадлежащих различным частям UI. Каждый из этих объектов содержит множество методов и свойств, к которым можно обращаться для работы с объектом и взаимодействия с ним.

На следующем рисунке показано одно представление этих объектов в иерархии объектной модели Word.

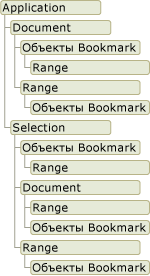


Рис. 1.1 – Иерархия объектной модели Word

## Анализ существующих решений

Данная задача осуществляет работу с файлами формата .docx, осуществляя поиск, преобразование, разархивирование и замену. В качестве аналогов данной программы можно рассматривать макросы, написанные пользователями на Visual Basic для личных целей, но официально таких программ не зарегистрировано.

## Постановка задачи

Используя динамическую библиотеку Microsoft.Office.Interop.Word.dll, следует разработать программу, которая будет генерировать файл реферата на основе пояснительной записки, формируя отчет о количестве изображений, таблиц, страниц и количестве ссылок, а также вносить данные о цели работы и записывать ключевые слова, то есть выполнять ту работу, которую можно автоматизировать.

# ПРОЕКТИРОВАНИЕ

## 2.1 Выбор и обоснование архитектуры

Программы – это основной элемент повсеместно используемых сотовых телефонов, а также комплексных систем управления воздушным движением. По сути, многих новшеств, которые мы теперь воспринимаем как данность – в том числе таких организаций, как eBay или Amazon – просто не существовало бы, если бы они не были построены на программном обеспечении. Даже традиционные организации в финансовом, общественном и торговом секторах в значительной мере зависят от программного обеспечения. На сегодняшний день трудно найти организацию, которая хоть каким–либо образом не была вовлечена в сферу программного обеспечения.

Архитектура – это базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы.

Архитектура ПО – это набор значимых решений по поводу организации системы программного обеспечения, набор структурных элементов и их интерфейсов, при помощи которых компонуется система, вместе с их поведением, определяемым во взаимодействии между этими элементами, компоновка элементов в постепенно укрупняющиеся подсистемы , а также стиль архитектуры который направляет эту организацию –– элементы и их интерфейсы, взаимодействия и компоновку.

Качественная архитектура является основой разработки любого проекта. Таковой можно считать только выгодную архитектуру, которая упрощает процесс создания программы и делает его более понятным. Таким образом можно сказать, что чем проще становится разработчику понять его задачу, тем лучше организована программа. Также результатом хорошо продуманной архитектуры является простота отладки программы, возможность вносить поправки и расширять возможности, не ломая при этом того, что было создано ранее. Архитектура программного обеспечения начинается с набора требований. Они могут быть выражены в форме диаграмм, блок–схем процесса, моделей или документированных списков задач эксплуатации, которые должно выполнять программное обеспечение. Обычно клиент или партнер также выражает менее точные требования, такие как внешний вид или способ работы определенных пользовательских интерфейсов для часто встречающихся задач. Требования также должны включать в себя информацию о существующем программном обеспечении, системах, оборудовании и сетях, с которыми будет взаимодействовать новое программное обеспечение; и другие факторы, такие как план развертывания и обслуживания, и, конечно же, доступный бюджет проекта[8].

В данном проекте будет использоваться трехуровневая архитектура. Разрабатываемая система разделается на три уровня, уровень представления или же интерфейс, уровень логики, которая включает в себя реализацию алгоритмов и уровень файловой системы.



Рисунок 2.1 – Архитектура данного проекта

## Диаграмма прецедентов

Функции, которые реализованны в проекте, можно визуализировать с помощью диаграммы прецедентов.



Рисунок 2.2 – Диаграмма прецедентов

В таблицах 2.1 – 2.5 описаны прецеденты представленные на рисунке 2.2

Таблица 2.1 – Прецедент ввода данных с клавиатуры

|  |  |
| --- | --- |
| Название прецедента | «Выбор документа пояснительной записки» |
| Описание | Данный прецедент позволяет вводить данные с клавиатуры для указания входных данных. |

Таблица 2.2 – Прецедент обработки

|  |  |
| --- | --- |
| Название прецедента | «Обработка информации» |
| Описание | Прецедент проверяет собирает данные из файла. |

Таблица 2.3 – Прецедент создания

|  |  |
| --- | --- |
| Название прецедента | «Генерация документа реферата» |
| Описание | Прецедент осуществляет генерацию документа реферата, куда вносятся данные о количестве страниц, картинок и т.д., согласно заданной постановке задачи. |

Таблица 2.4 – Прецедент сохранения

|  |  |
| --- | --- |
| Название прецедента | «Сохранение документа в указанном месте» |
| Описание | Прецедент реализует сохранение документа реферата по заданному пути и имени. |

Таблица 2.5 – Прецедент завершения работы

|  |  |
| --- | --- |
| Название прецедента | «Завершение работы с приложением» |
| Описание | Прецедент реализует закрытие программы после сохранения документа. |

## Диаграмма последовательности



Рисунок 2.3 – Диаграмма последовательности

## Проектирование графического интерфейса

В ходе разработки пользовательского интерфейса было реализовано оконное приложение с использованием WindowForms [9], которое соответствует всем описанным в техническом задании требованиям.

Приложение состоит из главной формы:

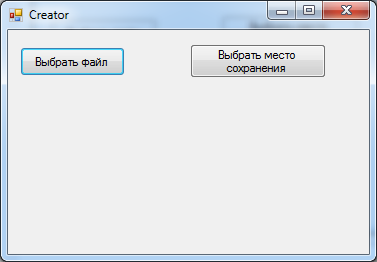


Рисунок 2.4 – Внешний вид главного окна приложения

Рабочая область содержит такие кнопки:

1. «Выбрать файл» – внесение данных о пути к файлу формата .docx с исходными данными.
2. «Выбрать место сохранения» – внесение данных

Рабочая область содержит следующие элементы:

1. «openFile» – элемент OpenFileDialog.
2. «saveFile» – элемент SaveFileDialog.

Пример работы:

1. Ввод пути к исходному документу

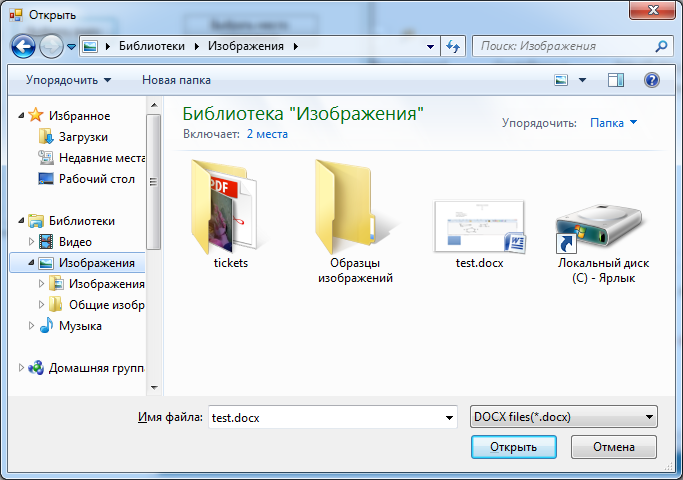


Рисунок 2.5 – Введение данных

1. Выбор папки для сохранения документа

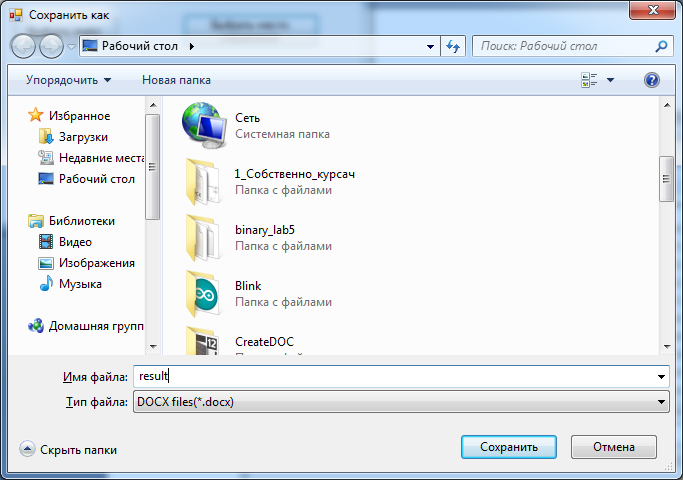


Рисунок 2.6 – Выбор папки для сохранения документа

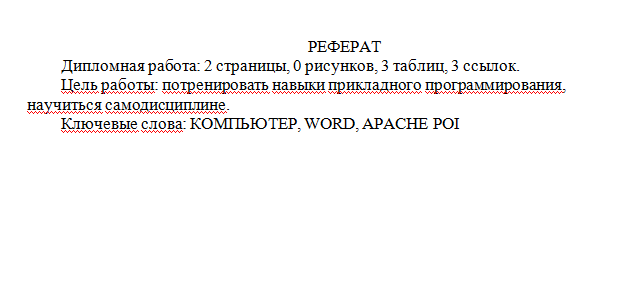


Рисунок 2.7 – Результат проверки сгенерированного документа

# РАЗРАБОТКА

## Диаграмма пакетов



Рисунок 3.1 – Диаграмма пакетов

На диаграмме пакетов, изображеной на рисунке 3.1 показана двухурованевая организация архитектуры. Класс Creator относится к уровню графического интерфейса. Класс SelectTable реализует подсчет таблиц и страниц, класс SelectPic реализует подсчет картинок, класс SelectLinks реализует подсчет ссылок, страниц, выбирает фразы про цель работы и ключевые слова, класс CreateDocx реализует генерацию документа на основе полученной информации.

## 3.2 Диаграмма классов

В ходе выполнения данной работы для решения поставленной задачи были разработаны классы Creator, CreateDocx, SelectLinks, SelectPic, SelectTables которые представлены на рисунке 3.2.

Описание:

1. Creator – класс для графического отображения ввода данных;
2. CreateDocx – класс для редактирования файла .docx;
3. SelectTable – класс для подсчета таблиц;
4. SelectPic – класс для подсчета изображений;
5. SelectLinks – класс для подсчета ссылок и др. информации.

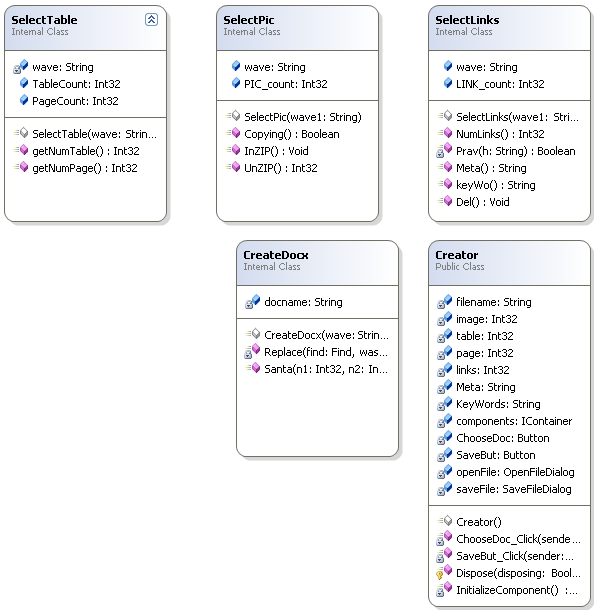


Рисунок 3.2 – Диаграмма классов

## Разработка и описание ключевых методов классов



Рисунок 3.3 – Общий алгоритм работы программы

В таблице 3.1 представлены поля и методы класса Creator

Таблица 3.1 – Таблица со списком полей и методов класса Creator.

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | filename(String) | Имя файла, в котором содержится пояснительная записка |
| 2. | image(Int32) | Счетчик изображений |
| 3. | table(Int32) | Счетчик таблиц |
| 4. | page(Int32) | Счетчик страниц |
| 5. | links(Int32) | Счетчик ссылок |
| 6. | Meta(String) | Строка цели работы |
| 7. | KeyWords(String) | Строка ключевых слов |
| 8. | ChooseDoc(Button) | Кнопка выбора документа пояснительной записки |
| 9. | SaveBut(Button) | Кнопка выбора места сохранения результирующего документа |
| 10. | openFile(OpenFileDialog) | Окно диалога открытия файла |
| 11. | saveFile(SaveFileDialog) | Окно диалога сохранения файла |
|  | *Методы* |  |
| 1. | ChooseDoc \_Click | Обработчик нажатия на клавишу ChooseDoc |
| 2. | SaveBut\_Сlick | Обработчик нажатия на клавишу SaveBut |

В таблице 3.2 представлены поля и методы класса CreateDocx

Таблица 3.2 – Таблица со списком полей и методов класса CreateDocx.

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | docname(String) | Имя файла, в который будут записываться результаты. |
|  | *Методы* | *Назначение* |
| 1. | Replace | Метод, в котором производится замещение ключевых слов данными |
| 2. | Santa | Метод, осуществляющий запись |

В таблице 3.3 представлены поля и методы класса SelectTable

Таблица 3.3 – Таблица со списком полей и методов класса SelectTable.

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | wave(String) | Имя файла, из которого берутся данные. |
| 2. | TableCount(Int32) | Счетчик таблиц |
| 3. | PageCount(Int32) | Счетчик страниц |
|  | *Методы* | *Назначение* |
| 1. | getNumTable | Метод, в котором производится подсчет таблиц |
| 2. | getNumPage | Метод, в котором производится подсчет страниц. |

В таблице 3.4 представлены поля и методы класса SelectPic

Таблица 3.4 – Таблица со списком полей и методов класса SelectPic.

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | wave(String) | Имя файла, из которого берутся данные. |
| 2. | Pic\_Count(Int32) | Счетчик изображений. |
|  | *Методы* | *Назначение* |
| 1. | Copying | Метод, в котором производится копирование исходного документа в директорию рядом с файлом .exe |
| 2. | InZIP | Метод, в котором производится переименование файла в файл с расширением .zip |
| 3. | UnZIP | Метод, в котором производится разархивирование файла и подсчет изображений в определенной директории |

В таблице 3.5 представлены поля и методы класса SelectLinks

Таблица 3.5 – Таблица со списком полей и методов класса SelectLinks.

|  |  |  |
| --- | --- | --- |
|  | *Поля* | *Назначение* |
| 1. | wave(String) | Имя файла, из которого берутся данные. |
| 2. | LINK\_Count(Int32) | Счетчик ссылок |
|  | *Методы* | *Назначение* |
| 1. | NumLinks | Метод, в котором производится подсчет ссылок |
| 2. | Prav | Метод, в котором производится определение ссылки из общего текста |
| 3. | Meta | Метод, в котором производится поиск цели работы |
| 4. | keyWo | Метод, в котором производится поиск ключевых слов |
| 5. | Del | Метод, в котором производится удаление мусора перед закрытием программы. |

# ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММЫ

## 4.1 Верификация

Верификацией называется проверка соответствия результатов отдельных этапов разработки программной системы требованиям и ограничениям, сформулированным для них на предыдущих этапах.

Согласно техническому заданию для реализации поставленной задачи был разработан проект, который позволяет:

1. Осуществлять ввод данных с клавиатуры.
2. Генерировать конечный документ.
3. Выдавать сообщение об ошибке в случае exception.
4. Cохранять конечный документ.

Задачу 1 выполняет прецедент: «Выбор документа пояснительной записки», результат представлен на рисунке 4.1

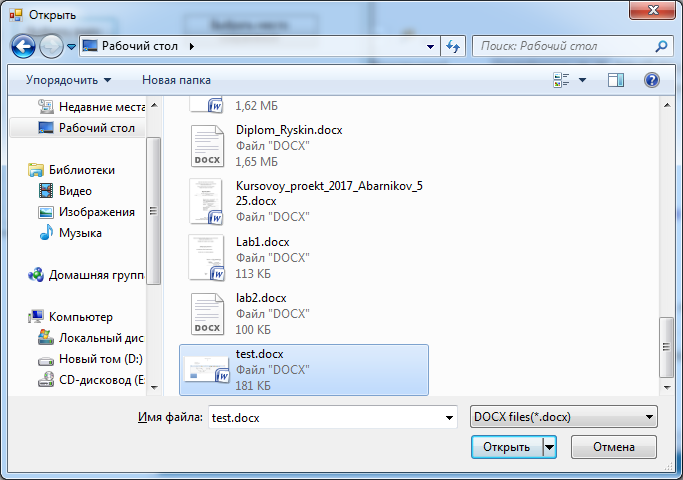


Рисунок 4.1 – Иллюстрация выполнения задачи 1

Результат выполнения задачи №2 представлен на рисунке 4.2

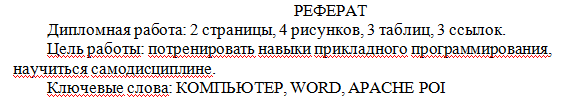


Рисунок 4.2 – Иллюстрация выполнения задачи 2

Результат выполнения задачи №3 представлен на рисунке 4.3

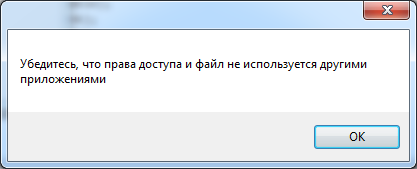


Рисунок 4.3 – Иллюстрация выполнения задачи 3

Результат выполнения задачи №4 представлен на рисунке 4.4

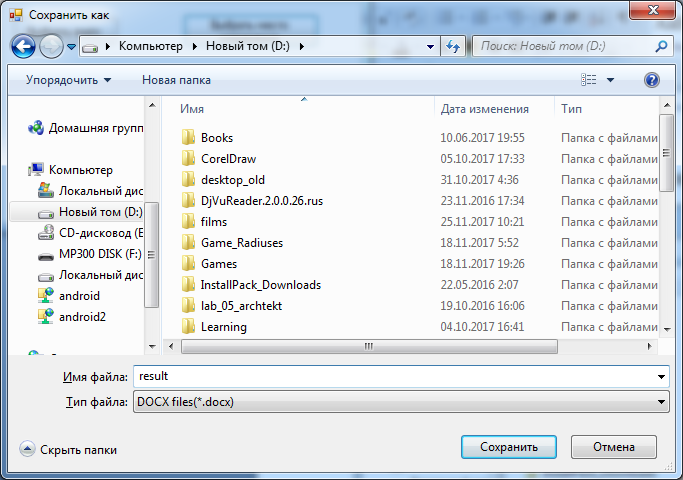


Рисунок 4.4 – Иллюстрация выполнения задачи 4

Поскольку эта записка является документом, который является корректным в качестве входных данных для программы, то результат работы данной программы представлен на рисунке 4.5

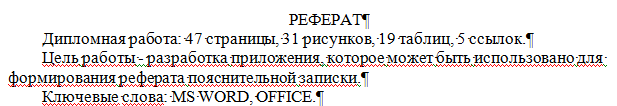


Рисунок 4.5 – Иллюстрация работы программы на данном документе

## 4.2 Тестирование программы

Тестирование программного обеспечения (Software Testing) – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

Поскольку основные аспекты тестов пройдены, следует пройтись по «запрещенным» конструкциям ветвлений, то есть тем, которые могут вызвать исключение. Например, попытка открыть файл, в котором вносятся изменения другим пользователем, или открытый в другой программе.

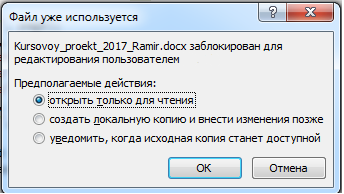


Рисунок 4.5 – Попытка открыть

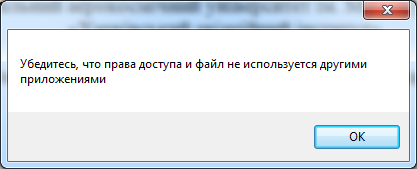


Рисунок 4.6 – Выдача предупреждения

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были изучены основные принципы работы с библиотекой Microsoft.Office.Interop.Word. Была проведена работа по созданию приложения для генерации документов. Программа верифицирована и протестирована. Тесты пройдены успешно.

В дальнейшем возможно усовершенствование программы:

1. Улучшение интерфейса;
2. Расширение списка форматов поддерживаемых документов.
3. Улучшение и оптимизация алгоритма по подсчетам.

# ПЕРЕЧЕНЬ ССЫЛОК

1. Текстовый файл [Электронный ресурс] / Режим доступа https://ru.wikipedia.org/wiki/Текстовый\_файл
2. Microsoft Word [Электронный ресурс] / Режим доступа https://ru.wikipedia.org/wiki/Microsoft\_Word
3. Общие сведения об объектной модели Word [Электронный ресурс] / Режим доступа https://msdn.microsoft.com/ru–ru/library/kw65a0we.aspx
4. .DOC [Электронный ресурс] / Режим доступа https://ru.wikipedia.org/wiki/.doc
5. SaveFileDialog [Электронный ресурс] / Режим https://metanit.com/sharp/windowsforms/4.20.php

# ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

**1 Введение**

**1.1 Наименование программы**

Наименование программы – Генерация документа реферата на основе пояснительной записки.

**1.2 Краткая характеристика области применения**

Данная программа предназначена для генерации документа реферата на основе пояснительной записки к курсовому или дипломному проекту.

**2 Основание для разработки**

**2.1 Основание для проведения разработки**

Основания для разработки – задание на курсовой проект по дисциплине «Объектно–ориентированное программирование» на тему «Генерация реферата на основе пояснительной записки»,выдано 24.01.2017.

**2.2 Наименование и условное обозначение темы разработки**

Наименование темы разработки – «Разработка программы для генерации реферата на основе пояснительной записки».

**3 Назначение разработки**

**3.1 Функциональное назначение программы**

Данная программа предоставляет возможность генерации реферата по пояснительной записке, подсчет графических элементов, таблиц, ссылок и страниц.

**4** **Эксплуатационное назначение**

Программный продукт был разработан средствами Windows Forms в среде разработки Visual Studio 2017. Программа предназначена для работы на ОС Windows и не является кроссплатформенной, изменения в графическом отображении возможны.  
Программа не нуждается в обслуживании. Программа распространяется как откомпилированный файл \*.exe. Сбой или отказ работы программы может привести к зависанию самой программы.

**5 Требования к программе или программному изделию**

**5.1 Требования к функциональным характеристикам**

**5.1.1 Требования к составу выполняемых функций**

Программа должна выполнять следующие функции:

а) осуществлять ввод данных с клавиатуры

б) задавать имя для документа;

в) генерировать конечный документ;

г) сохранять конечный документ.

**5.1.2 Требования к организации входных данных**

Файл должен быть формата .docx, который открывается корректно при помощи официального продукта компании Microsoft Office Word.

**5.1.3 Требования к организации выходных данных**

В качестве выходных данных должен создаваться файл формата .docx.

**5.2 Требования к составу и параметрам технических средств**

В состав технических средств должен входить x86–совместимый компьютер, используемый в качестве рабочей станции и включающий в себя:

а) процессор с тактовой частотой, МГц – 400, не менее;

б) материнскую плату с FSB, МГц – 330, не менее;

в) оперативную память объемом, Мб –1024, не менее;

г) свободное дисковое пространство для хранения программы, Мб – 500, не менее;

е) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

ж) клавиатуру;

з) мышь.

**5.3 Требования к информационной и программной совместимости**

**5.3.1 Требования к информационным структурам и методам решения**

Требования к информационным структурам на входе и выходе, а также к методам решения не предъявляются.

**5.3.2 Требования к исходным кодам и языкам программирования**

Для разработки программы в качестве языка программирования нужно использовать язык C# поддерживающий спецификацию компании производителя Microsoft, а за графический интерфейс отвечает Windows Forms. Для проектирования необходимо использовать среду разработки Microsoft Visual Studio версии 2010 и выше.

Исходные тексты программ должны содержать достаточное количество комментариев и быть доступными для любого читающего. Необходимо использовать осмысленные и корректные названия, объясняющие свое назначение.

**5.3.3 Требования к программным средствам, используемым программой**

Для функционирования программы на компьютере должно быть установлено следующее ПО:

* ОС Windows 7 и выше;
* платформа .NET Framework 4.5 и выше;
* Microsoft Visual Studio версии 2015 и выше.

**5.3.4 Требования к защите информации и программ**

Программа не имеет доступ к конфиденциальной информации, поэтому никакие требования к защите информации и программы не предъявляются.

**6 Требования к программной документации**

**6.1 Предварительный состав программной документации**

В результате разработки программы должна быть представлена следующая программная документация:

а) техническое задание;

б) схемы алгоритмов и данных;

в) текст программы;

г) результаты тестирования;

д) пояснительная записка;

е) руководство пользователя.

Кроме программного обеспечения на диске обязательно должна находиться пояснительная записка в электронном виде, включающая в себя весь комплект документации, предусмотренный в п. 5.1 настоящего технического задания, а также слайды презентации.

**7 Стадии и этапы разработки**

**7.1 Стадии разработки**

Разработка должны быть проведена в 4 этапа:

1. Проектирование.

2. Разработка пользовательского интерфейса.

3. Разработка проекта в целом.

4. Разработка руководства пользователя.

**7.2 Этапы разработки**

На этапе проектирования необходимо определить тему работы и разработать техническое задание. На этапе разработки пользовательского интерфейса необходимо разработать интерфейс программы. На этапе разработки проекта в целом необходимо разработать проект программы и провести ее тестирование. В заключительном этапе необходимо разработать руководство пользователя и подготовить программу к использованию.

# ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ

Creator.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace CreateDOC

{

public partial class Creator : Form

{

string filename;

int image;

int table;

int page;

int links;

string Meta;

string KeyWords;

public Creator()

{

InitializeComponent();

openFile.Filter = "DOCX files(\*.docx)|\*.docx";

saveFile.Filter = "DOCX files(\*.docx)|\*.docx";

}

private void ChooseDoc\_Click(object sender, EventArgs e)

{

if (openFile.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

filename = openFile.FileName;

}

private void SaveBut\_Click(object sender, EventArgs e)

{

if (saveFile.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

string DocName = saveFile.FileName;

try

{

File.Delete("daoc.docx");

File.Delete("daoc.zip");

Directory.Delete("libr", true);

}

catch { }

try

{

SelectPic maer = new SelectPic(filename);

// вытянуть количество рисунков

maer.Copying();

maer.InZIP();

image = maer.UnZIP();

//вытянуть кол-во таблиц

SelectTable rt = new SelectTable(filename);

table = rt.getNumTable();

//вытянуть кол-во страниц

page = rt.getNumPage();

//вытянуть кол-во ccылок

SelectLinks dert = new SelectLinks(filename);

links = dert.NumLinks();

//вытянуть цель работы

Meta = dert.Meta();

KeyWords = dert.keyWo();

dert.Del();

string pathParent = "parene.docx";

if (File.Exists(DocName))

{

File.Delete(DocName);

}

File.Copy(pathParent, DocName);

// сохраняем текст в файл

CreateDocx merr = new CreateDocx(DocName);

merr.Santa(page, image, table, links, KeyWords, Meta);

}

catch(Exception p){

MessageBox.Show("Убедитесь, что права доступа и файл не используется другими приложениями");

}

try

{

File.Delete("daoc.docx");

File.Delete("daoc.zip");

Directory.Delete("libr", true);

}

catch { }

Application.Exit();

}

}

}

CreateDocx.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Word = Microsoft.Office.Interop.Word;

namespace CreateDOC

{

class CreateDocx

{

string docname;

public CreateDocx(string wave)

{

docname = wave;

}

private void Replace(Word.Find find, string was, string will)

{

find.ClearFormatting();

find.Replacement.ClearFormatting();

find.Text = was;

find.Replacement.Text = will;

object missing = Type.Missing;

object replaceAll = Microsoft.Office.Interop.Word.WdReplace.wdReplaceAll;

find.Execute(ref missing, ref missing, ref missing, ref missing, ref missing,

ref missing, ref missing, ref missing, ref missing, ref missing,

ref replaceAll, ref missing, ref missing, ref missing, ref missing);

}

/// /// <summary>

/// И, наконец, замена

/// </summary>

/// <param name="n1">КОличество страниц</param>

/// <param name="n2">КОличество рисунков</param>

/// <param name="n3">КОличество таблиц</param>

/// <param name="n4">КОличество ссылок</param>

/// <param name="keyWord">Ключевые слова</param>

/// <param name="Meeta">Цель рабты</param>

public void Santa(int n1, int n2, int n3, int n4, string keyWord, string Meeta){

string image = Convert.ToString(n2);

string page = Convert.ToString(n1);

string table = Convert.ToString(n3);

string link = Convert.ToString(n4);

Microsoft.Office.Interop.Word.Application app = new Microsoft.Office.Interop.Word.Application();

app.Visible = false;

app.Documents.Open(docname);

Microsoft.Office.Interop.Word.Find find = app.Selection.Find;

Replace(find, "num\_page", page);

Replace(find, "num\_pic", image);

Replace(find, "num\_tables", table);

Replace(find, "num\_links", link);

Replace(find, "num\_links", link);

Replace(find, "CelWork", Meeta);

Replace(find, "KeyWords", keyWord);

object saveOptionsObject = Word.WdSaveOptions.wdSaveChanges;

app.Quit(saveOptionsObject, Type.Missing, Type.Missing);

}

}

}

SelectLinks.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace CreateDOC

{

class SelectLinks

{

public string wave;

public int LINK\_count;

public SelectLinks(string wave1){

wave = wave1;

wave += ".txt";

}

public int NumLinks()

{

string line;

string s = "Перечень";

string s1 = "ПЕРЕЧЕНЬ";

int o = 1;

string s2, s3;

System.IO.StreamReader file = new System.IO.StreamReader(wave, System.Text.Encoding.Default);

while ((line = file.ReadLine()) != null)

{

if ((line.Contains(s) || line.Contains(s1))&&Prav(line))

{

while((s2=file.ReadLine()) != null){

s3 =Convert.ToString(o)+".";

//if(s2.IndexOf())

if (s2.IndexOf(s3) == 0)

{

o++;

continue;

}

else

{

file.Close();

return o-1;

}

}

file.Close();

return 0;

}

}

file.Close();

return 0;

}

bool Prav(string h)

{

string ex;

for (int i = 0; i < 10; i++)

{

ex = Convert.ToString(i);

if (h.Contains(ex))

return false;

}

return true;

}

public string Meta()

{

string line;

System.IO.StreamReader file = new System.IO.StreamReader(wave, System.Text.Encoding.Default);

while ((line = file.ReadLine()) != null)

{

if (line.Contains("Цель работы"))

{

file.Close();

return line;

}

}

file.Close();

return "Цель работы - ";

}

public string keyWo()

{

string line;

System.IO.StreamReader file = new System.IO.StreamReader(wave, System.Text.Encoding.Default);

while ((line = file.ReadLine()) != null)

{

if (line.Contains("Ключевые слова"))

{

file.Close();

return line;

}

}

file.Close();

return "";

}

public void Del()

{

File.Delete(wave);

}

}

}

SelectPic.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.IO.Compression;

namespace CreateDOC

{

class SelectPic

{

public string wave;

public int PIC\_count;

public SelectPic(string wave1){

wave = wave1;

}

public bool Copying()

{

try

{

string MotherLand = "daoc.docx";

File.Copy(wave, MotherLand);

return true;

}

catch

{

return false;

}

}

public void InZIP(){

File.Move("daoc.docx", "daoc.zip");

}

public int UnZIP()

{

try

{

ZipFile.ExtractToDirectory("daoc.zip", "libr");

PIC\_count = new DirectoryInfo(@"libr\word\media").GetFiles().Length; ///vk.com/lerka\_gorbatko

File.Move("daoc.zip", "daoc.docx");

return PIC\_count;

}

catch

{

return 0;

}

}

}

}

SelectTable.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Word = Microsoft.Office.Interop.Word;

namespace CreateDOC

{

class SelectTable

{

string wave;

public int TableCount;

public int PageCount;

public SelectTable(string wave)

{

this.wave = wave;

}

public int getNumTable(){

Word.Application word = new Word.Application(); //создаем COM-объект Word

word.Visible = false;

Word.Document doc = word.Documents.Open(wave);

TableCount=doc.Tables.Count;

doc.Close();

word.Quit();

return TableCount;

}

public int getNumPage()

{

Word.Application word = new Word.Application(); //создаем COM-объект Word

word.Visible = false;

Word.Document doc = word.Documents.Open(wave);

Word.WdStatistic pages = Word.WdStatistic.wdStatisticPages;

object Format = Word.WdSaveFormat.wdFormatUnicodeText;

PageCount = doc.ComputeStatistics(pages);

wave += ".txt";

doc.SaveAs(wave, Format);

doc.Close();

((Word.Application)word).Quit();

return PageCount;

}

}

}

# ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Назначение программы

Программа представляет собой удобный и интуитивно понятный пользовательский интерфейс для создания документа по шаблону.

1. Системные требования

В состав технических средств должен входить x86–совместимый компьютер, используемый в качестве рабочей станции и включающий в себя:

а) процессор с тактовой частотой, МГц – 400, не менее;

б) материнскую плату с FSB, МГц – 330, не менее;

в) оперативную память объемом, Мб –1, не менее;

г) свободное дисковое пространство для хранения программы, Мб – 1, не менее;

е) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

ж) клавиатуру;

з) мышь.

1. Выполнение программы
2. Запустите приложение «CreateDOC.exe».

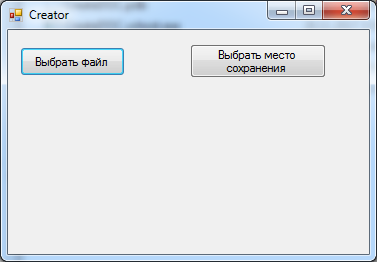


Рисунок Р.П.1 – Внешний вид приложения.

1. При нажатии кнопки «Выбрать файл» на экран будет выведено окно Проводника с предложением ввести имя документа для открытия и считывания информации.
2. При нажатии кнопки «Выбрать место сохранения» на экран будет выведено окно Проводника с предложением ввести имя документа для его создания, генерации и сохранения.
3. Закрыть программу можно нажав крестик в правом верхнем углу.

# ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ

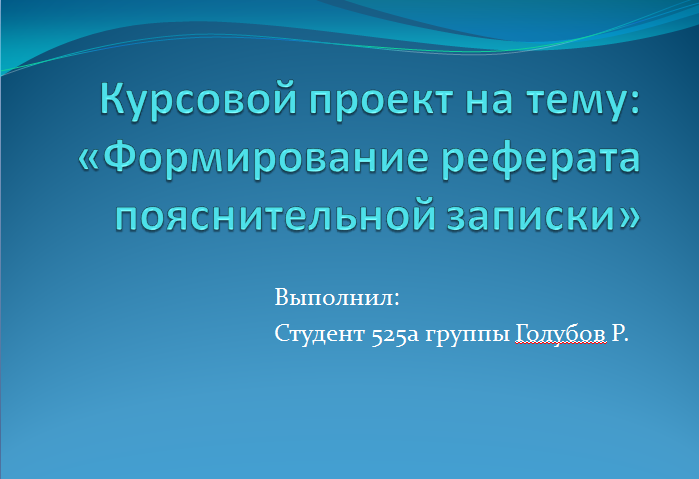


Рисунок П.1 – Слайд 1

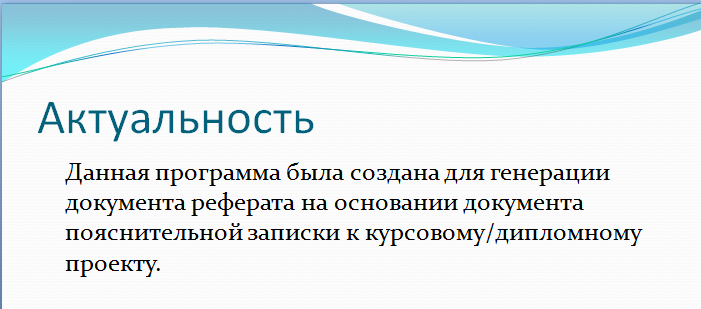


Рисунок П.2 – Слайд 2

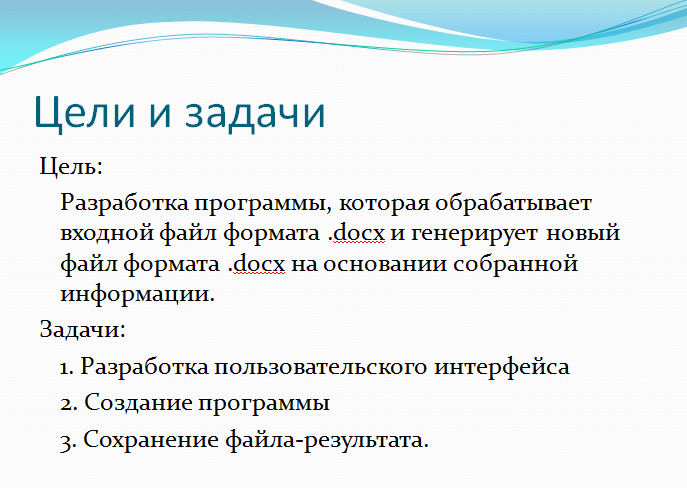


Рисунок П.3 – Слайд 3



Рисунок П.4 – Слайд 4



Рисунок П.5 – Слайд 5



Рисунок П.6 – Слайд 6



Рисунок П.7 – Слайд 7

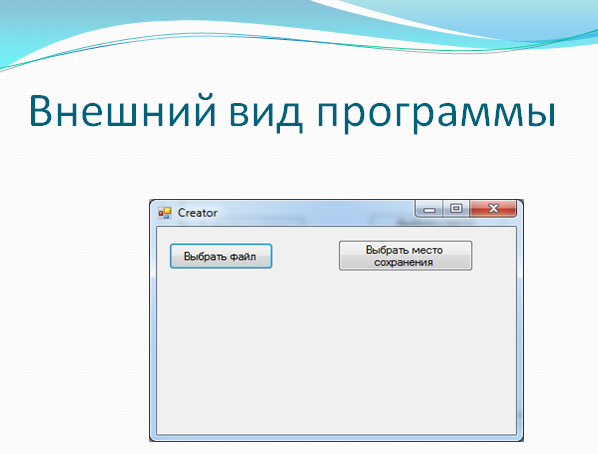


Рисунок П.8 – Слайд 8

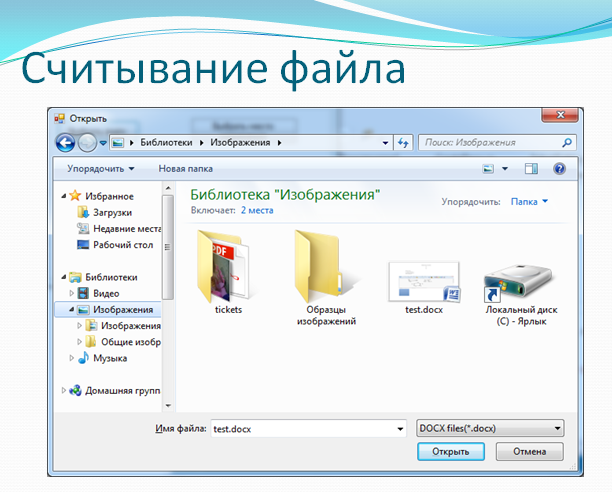


Рисунок П.9 – Слайд 9

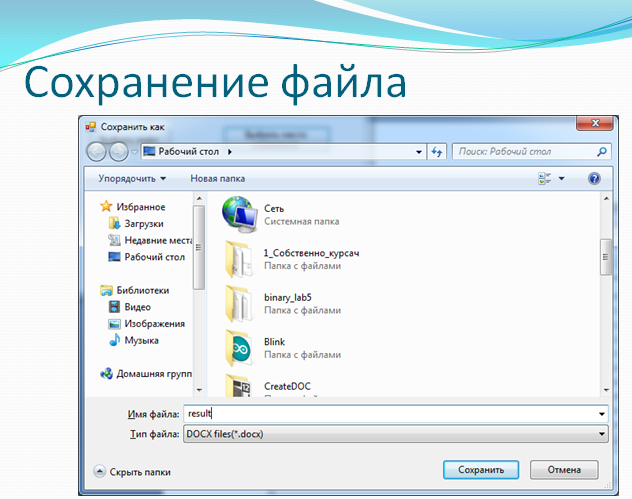


Рисунок П.10 – Слайд 10

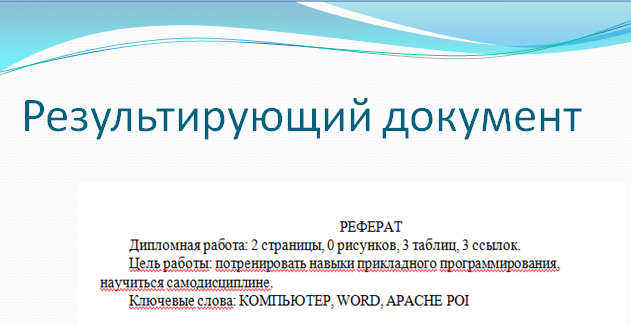


Рисунок П.11 – Слайд 11

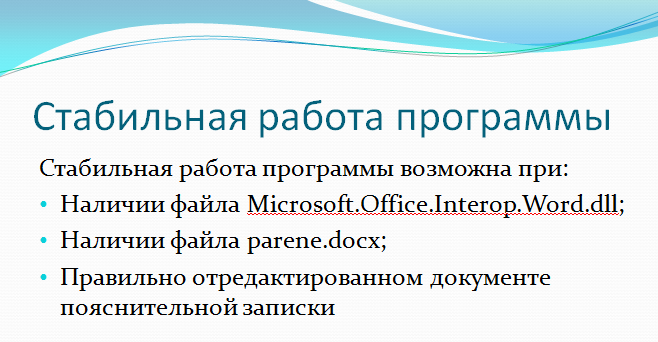


Рисунок П.12 – Слайд 12

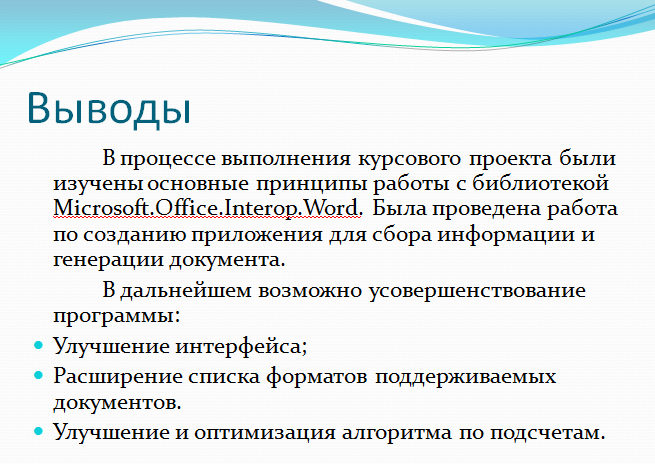


Рисунок П.13 – Слайд 13