МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

02.03.03 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

БАКАЛАВРСКАЯ РАБОТА

на тему **“Разработка конвертера контентов для системы дистанционного обучения”**

Студент А.С. Нестеров _____

Руководитель Н.И. Лиманова _____

Консультант по аннотации К.А. Селиверстова _____

Допустить к защите
Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____

«_____» _____ 2017 г.

Тольятти 2017

АННОТАЦИЯ

Тема выпускной квалификационной работы: “Разработка конвертера контентов для системы дистанционного обучения”.

Цель выпускной квалификационной работы – разработка приложения для создания презентаций на основе текстовых документов.

Разработка приложения осуществлялась для отдела сопровождения дистанционного обучения ЦНИТ ТГУ, с целью автоматизировать процесс создания презентаций, что позволит сэкономить время и облегчить труд работников отдела.

В ходе выполнения ВКР было проанализировано предприятие, для которого разрабатывалось приложение, так же были проанализированы исходные данные и требования к создаваемым презентациям. Далее были выбраны средства реализации, сформированы функциональные требования и разработана диаграмма классов. На последнем этапе было реализовано и протестировано приложения. Результаты тестирования показали, что разработанное приложение соответствует всем предъявляемым требованиям и, следовательно, позволяет упростить процесс создания презентации.

При разработке приложения использовались такие инструменты разработки как объектно-ориентированный язык программирования C#, среда разработки Microsoft Visual Studio, для взаимодействия с документами использовалась технология COM Interop.

Данная выпускная квалификационная работа состоит из пояснительной записки на 49 страниц, введения на 2 страницы, включая 27 рисунков, 5 таблиц, списка литературы из 22 источников, в том числе 5 источников на иностранном языке и 2 приложений.

ABSTRACT

The theme of graduation work: “Developing a content converter for a distance learning system”.

The aim of the work is development of an application for creating presentations based on text documents.

The application development was carried out for the department of distance learning support of the Center for new information technologies of Togliatti State University, in order to automate the process of creating presentations, which will save time and ease the work of the department's employees.

During the performance of the graduation work, the enterprise for which the application was developed was analyzed, as well as the initial data and requirements for the presentations being created. Further, the means of implementation were chosen, functional requirements were formed, and a class diagram was developed. At the last stage, applications were implemented and tested. The test results showed that the developed application meets all the requirements, and, therefore, allows to simplify the process of creating a presentation.

When developing the application, development tools such as the object-oriented programming language C#, the Microsoft Visual Studio development environment, and COM Interop technology were used to interact with documents.

The graduation work consists of an explanatory note on pages 49, introduction 2 pages, including 27 figures, 5 tables, the list of 22 references including 6 foreign sources and 2 appendices.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Образование является одной из важнейших сфер жизни современного общества. Образования несет важные функции такие как: социальная, приобретение знаний, умений и т.д. Если речь идёт о высшем образовании, то в этом случае основная цель – подготовка высококвалифицированных специалистов [2]. На данный момент основными формами обучения считаются очная и заочная, но в нашу высокотехнологическую эпоху информационные и телекоммуникационные технологии проникают практически во все сферы жизни общества, образование не является исключением. В связи с этим популярность набирает дистанционное обучение [18].

Дистанционное обучение предполагает взаимодействие между преподавателем и студентом на расстоянии с помощью информационных и телекоммуникационных технологий. Реализация дистанционного обучения через сеть интернет осуществляется через систему дистанционного обучения, одной из её функций является хранение образовательного контента, созданного преподавателем. Однако расширение файла, в котором содержится контент, является не всегда удобным для обучающегося или может не поддерживаться системой дистанционного обучения. В этом случае документ необходимо преобразовать в приемлемый формат.

Объект выпускной квалификационной работы – процесс разработки приложений для работы с документами Microsoft Office.

Предмет исследования выпускной квалификационной работы – приложение для обработки документов форматов docx и pptx.

Цель выпускной квалификационной работы – разработка приложения для создания презентаций на основе текстовых документов.

Исходя из цели работы, необходимо было решить следующие задачи:

- проанализировать исходные данные и требования к создаваемым презентациям;
- выбрать средства для реализации приложения;

- сформировать основные требования и разработать структуру разрабатываемого приложения;
- реализовать основные классы приложения;
- реализовать графический интерфейс пользователя;
- провести тестирование с целью выявления ошибок в работе приложения.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка литературы и приложений.

Во введении описывается актуальность проводимого исследования, формируется цель и ставятся задачи, которые необходимо решить для достижения цели.

В первой главе анализируется предприятие, для которого должно быть разработано программное обеспечение, анализируются исходные данные, а также рассматриваются требования к создаваемым презентациям.

Во второй главе выбирается язык программирования, среда разработки, анализируются средства взаимодействия с документами Microsoft Office и исходя из анализа выбирается наиболее подходящее средство. Формируются требования к разрабатываемому программному обеспечению, а также проектируются основные классы.

В третьей главе выполняется реализации основных классов приложения, графического интерфейса и проводится тестирование.

В заключении приводятся основные выводы по работе, достигнутые в ходе выполнения выпускной квалификационной работы.

Глава 1. Постановка задачи и анализ исходных текстовых документов

1.1 Анализ деятельности предприятия

Центр новых информационных технологий (ЦНИТ) Тольяттинского Государственного Университета был основан в декабре 2000 года. Фрагмент организационной структуры ТГУ представлен на рисунке 1.1.

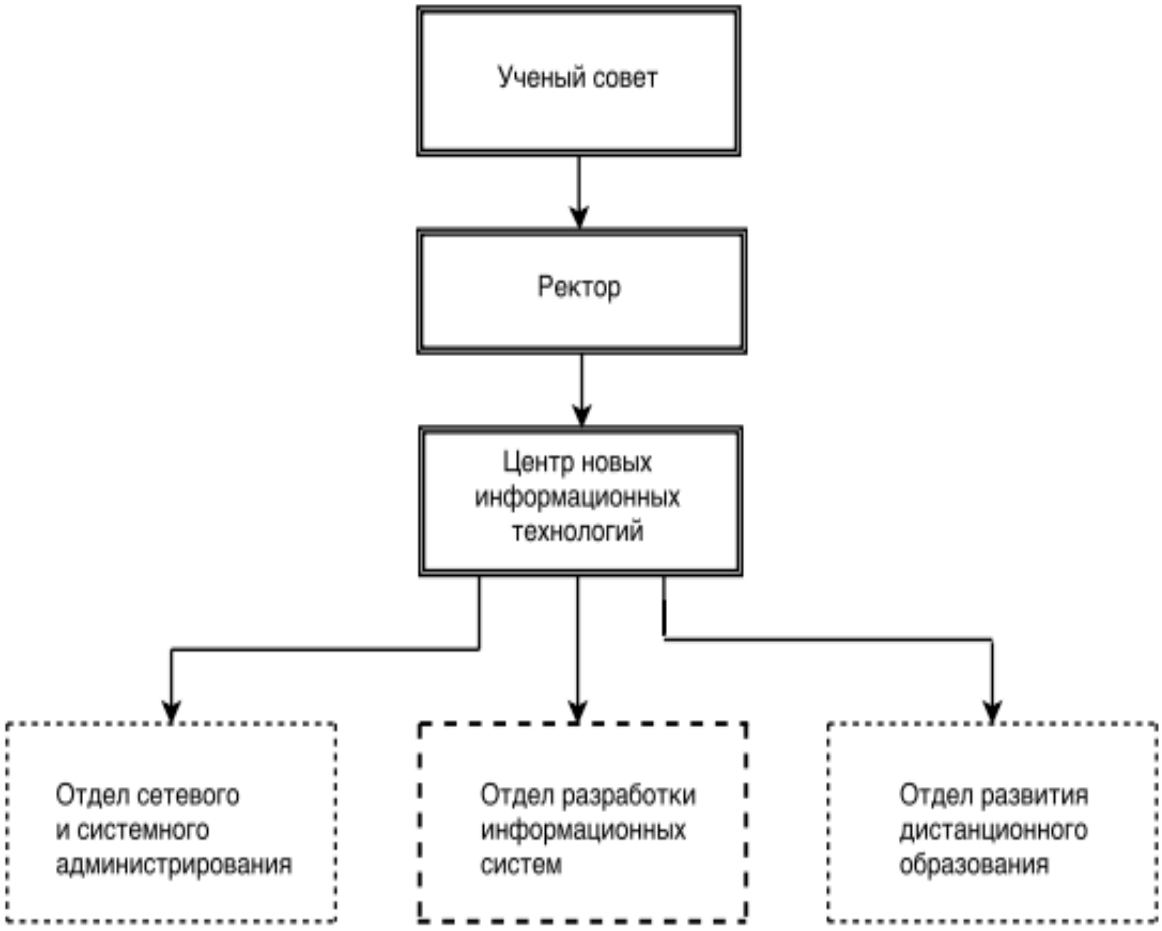

Рисунок 1.1 – Фрагмент структуры ТГУ

ЦНИТ специализируется в области информационных технологий, информатизации, автоматизации образовательной, административной и других сфер деятельности университета.

ЦНИТ включает в себя следующие отделы: отдел разработки информационных систем, отдел сетевого администрирования, отдел

менеджмента качества и оптимизации бизнес-процессов, отдел сопровождения корпоративной ERP-системы, технический отдел, отдел развития дистанционного обучения [10].

Задачами отдела сопровождения дистанционного обучения являются:

- осуществление планирования, а так же контроль реализации учебных курсов с использованием современных образовательных (в т.ч. дистанционных) технологий в университете;
- упаковка готовых материалов в формат «Росдистант»;
- поддержка и развитие системы дистанционного обучения;
- предоставление IT-услуги пользователям.

В ходе данной выпускной квалификационной работы необходимо разработать программное обеспечение для автоматизации деятельности отдела сопровождения дистанционного, в частности, создание презентаций на основе текстового документа, который содержит учебный материал.

1.2 Постановка задачи на разработку приложения

С развитием информационных и телекоммуникационных технологий дистанционная форма обучения становится одной из самых перспективных форм. В отличие от традиционного заочного обучения, дистанционное предполагает постоянное общение преподавателя и учащегося. Такое общение позволяет учащемуся консультироваться у преподавателей, а также дает возможность преподавателям наблюдать за ходом усвоения материала и организовывать обучение на основе индивидуального подхода. Помимо этого, дистанционное обучение, основанное на использовании сети Интернет, позволяет получить доступ ко всем уровням образования, вне зависимости от места проживания человека, его финансовых и физических возможностей или в силу других причин. Для реализации обучения через Интернет используются так называемые системы дистанционного обучения. Система дистанционного обучения – это комплекс программно-технических средств, методик и организационных мероприятий, которые позволяют

обеспечить доставку образовательной информации учащимся по компьютерным сетям общего пользования, а также осуществить проверку знаний, полученных в рамках курса обучения конкретным учащимся [7]. Таким образом, система дистанционного обучения позволяет хранить образовательный контент, а также контролировать успеваемость студентов.

Электронный образовательный контент или электронный учебник – это содержимое, которое загружается в систему дистанционного обучения, оно предназначено для непосредственного восприятия пользователем с целью обучения или ориентации в учебном процессе. Электронный учебник может представляться в виде презентации [11]. Материал для электронного учебника создается преподавателем в виде файлов с расширением docx (Microsoft Word). Далее текстовый документ передается сотрудникам отдела сопровождения дистанционного обучения, а работники отдела в свою очередь загружают материалы в систему дистанционного обучения.

Но так как сотрудники отдела сопровождения дистанционного обучения получают образовательный контент в файлах формата docx, им необходимо конвертировать его в презентацию (формат pptx). Чтобы сэкономить время и облегчить труд работников отдела, процесс преобразование форматов можно автоматизировать, для этого необходимо разработать приложение, которое будет осуществлять конвертацию автоматически.

1.3 Требования к создаваемым презентациям

Как уже было сказано, результатом работы программы должна стать презентация. К оформлению будущих презентаций налагается ряд различных требований.

Тема оформления презентаций строго определенна, она (тема) представляет собой белый фон с логотипом «Росдистанта». На рисунке 1.2 представлен пример слайда.

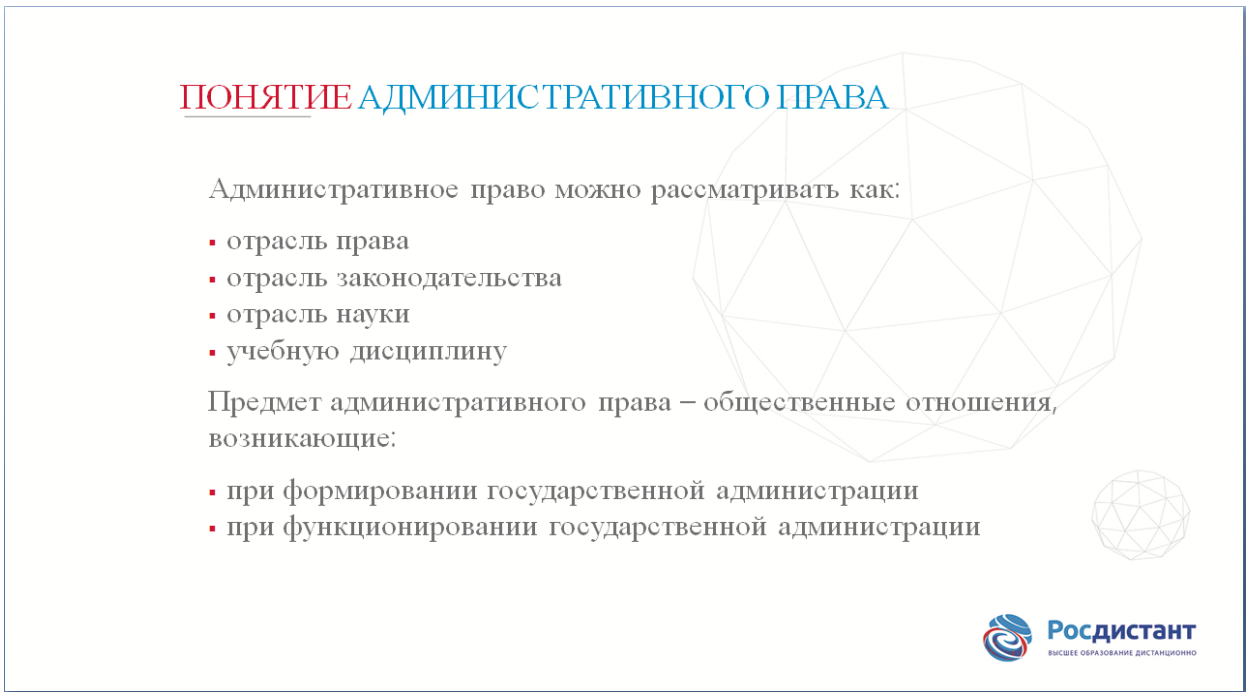

Рисунок 1.2 – Пример слайда

Требования к шрифту, цвету текста презентаций:

- цвет заголовка – синий. За исключение первого слова. Первое слово должно быть красного цвета;
- размер шрифта заголовка должен быть равен 27;
- размер шрифта текста основной фигуры должен быть равен 24;
- шрифт для всего текста – «Roboto Cn»;
- цвет текста серый (RGB (112,112,112)). Если текст в текстовом документе выделен «жирным», то на слайде этот текст будет тёмно серый (RGB (63,63,63)).
- для маркированного списка применяется символ – квадрат (Юникод – 0x25AA). Цвет символа – красный.

Требования к размещению таблиц следующие:

- текст таблицы аналогичен всему тексту презентации;
- размер текста в таблице равен 22;
- границы таблицы – синего цвета;

- если на слайде присутствует только таблица, она выравнивается по центру слайда, иначе она располагается в тексте.

  Пример слайда с таблицей представлен на рисунке 1.3.

Рисунок 1.3 – Пример слайда с таблицей

Требования для рисунков: если рисунок один (без других объектов), то он, также как и таблица, выравнивается по центру слайда, иначе помещается в фигуру «овал» (рисунок 1.4).

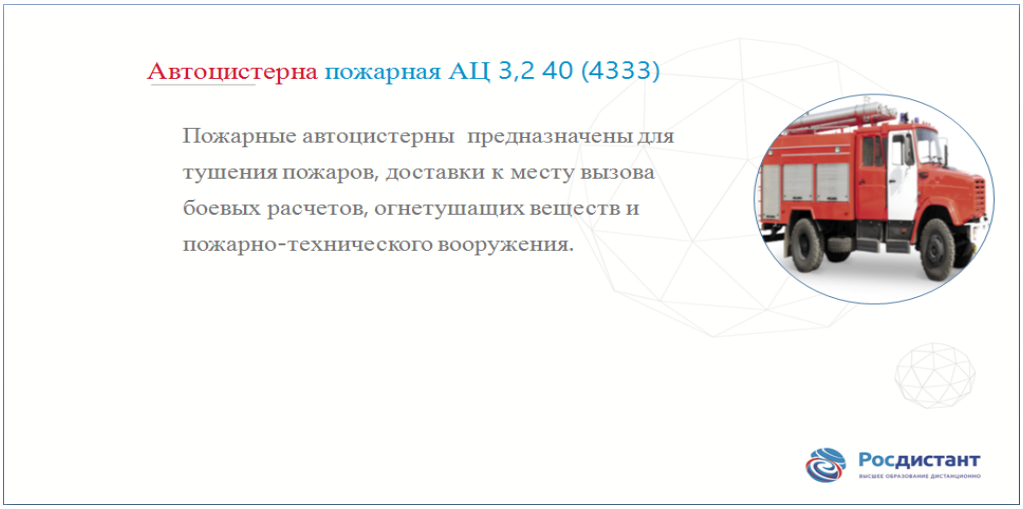

Рисунок 1.4 – Пример слайда с рисунком

Пример слайда с формулами представлен на рисунке 1.5. Требования к математическим формулам: формулы должны располагаться в тексте и размер шрифта формул соответствует размеру шрифта основного текста.

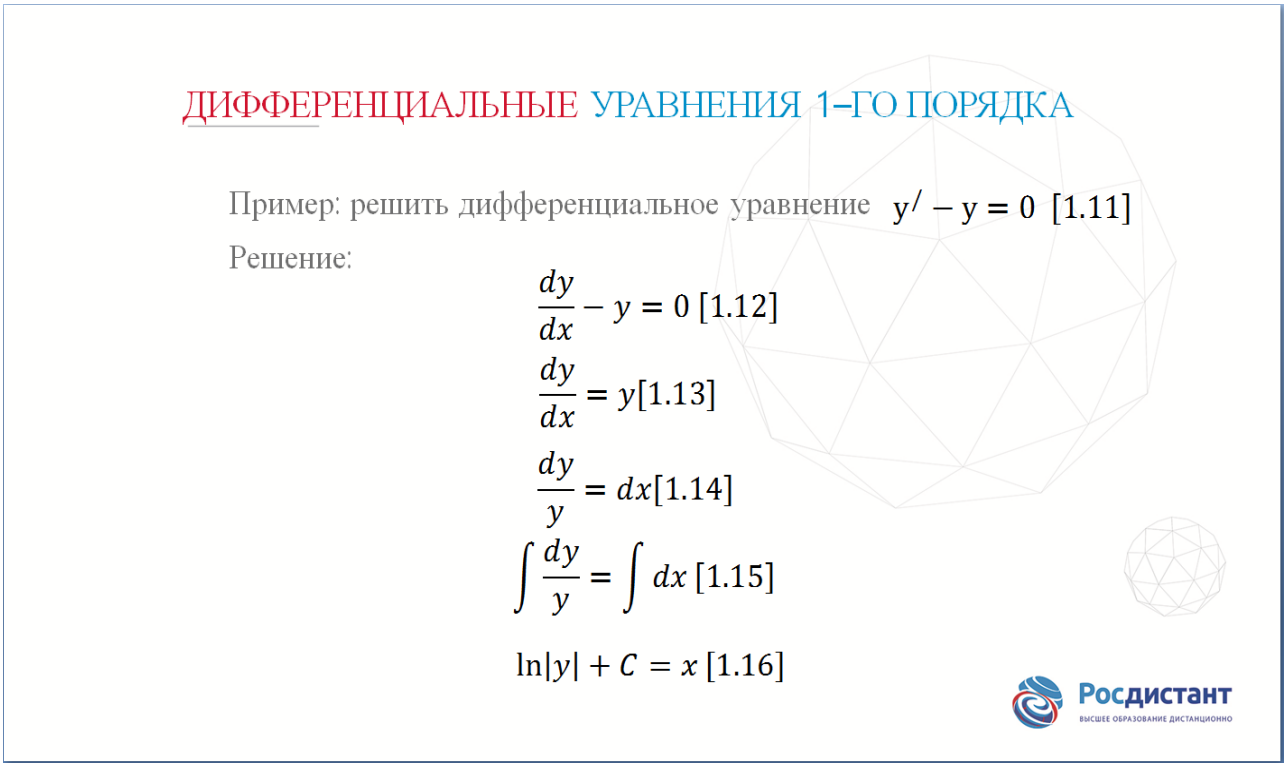

Рисунок 1.5 – Пример слайда с формулами

Требования к математическим формулам: формулы должны располагаться в тексте и размер шрифта формул соответствует размеру шрифта основного текста

1.4 Анализ исходных данных

Так как презентации создаются на основе текстового документа, то необходимо знать, по какому принципу (шаблону) разрабатываются исходные документы. Далее будут рассмотрены возможные шаблоны, по которым создаются слайды.

Один из вариантов шаблона, по которому будет создаваться слайд, представлен на рисунке 1.6. Исходный документ формата docx, являющийся основой для презентации, разделен на блоки (шаблоны новых слайдов). Блок

начинается с заголовка (номер слайда). После заголовка, определяющего начало нового слайда, следует слово – «заголовок». Текст, следующий после слова «заголовок», представляет заголовок будущего слайда. После слова «заголовок» следует «текст на слайде», под «текст на слайде» находится основное информационное наполнения, которое должно быть перенесено на слайд.

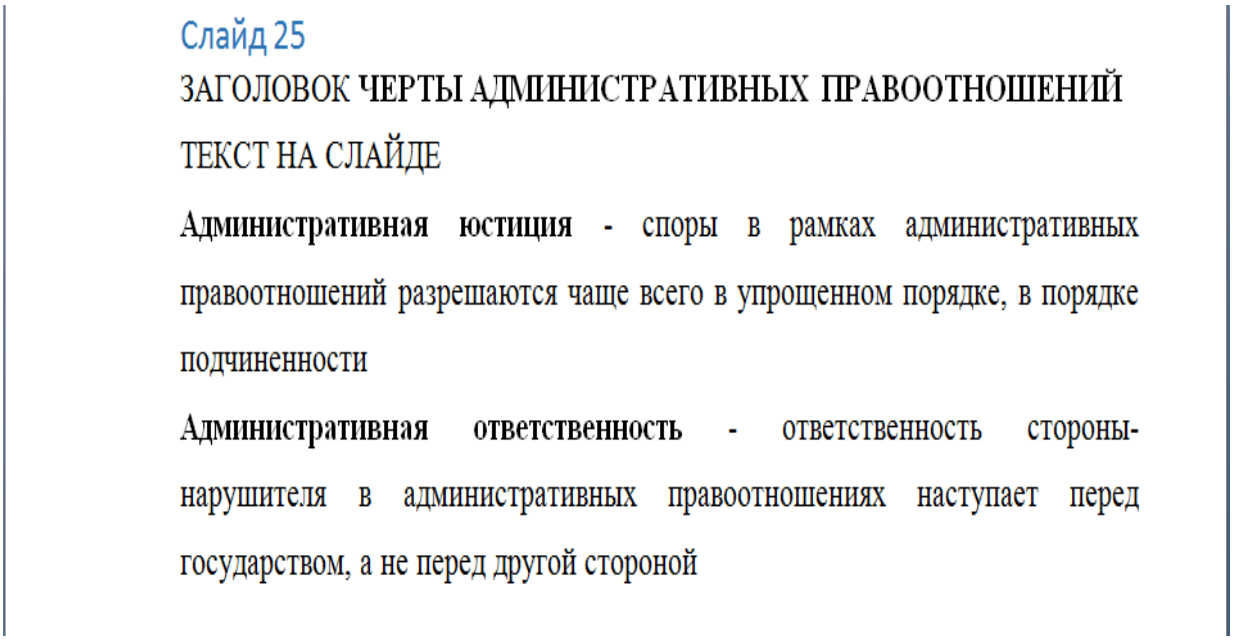

Рисунок 1.6 – Шаблон будущего слайда, содержащий только текст

Как видно из рисунка 1.6, шаблон содержит только текст. Следовательно, основным наполнением будущего слайда, созданного по данному шаблону, будет только текст.

Кроме текста, на слайд может быть помещена, например, таблица. На рисунке 1.7 представлен шаблон, который содержит таблицу. Возможны варианты, когда шаблон содержит таблицу вместе с текстом или вместе с математическими формулами. Также важно учитывать, что в самой таблице может располагаться не только текст, но и формулы, рисунки и другие объекты. Если не учесть тот факт, что в таблице может быть не только текст, то возможны проблемы при её (таблицы) переносе на слайд. Следующий вариант шаблона – это шаблон, содержащий рисунок. Данный шаблон

представлен на рисунке 1.8. По аналогии с шаблоном, который содержит таблицу, шаблон, содержащий рисунок, может содержать как один рисунок, так и рисунок одновременно с текстом или формулами, в зависимости от ситуации требования к оформлению слайдов будут отличаться.

Слайд 10

ЗАГОЛОВОК **АДМИНИСТРАТИВНОЕ ПРАВО И ИНЫЕ ОТРАСЛИ**

ТЕКСТ НА СЛАЙДЕ (таблица)

Административное и Конституционное право		
Предмет	вертикальные властные правоотношения нисходящего подчинения	властные отношения восходящего подчинения
Метод	императивный, властный	Реординационный
Административное и Гражданское право		
Предмет	имущественные отношения - в обеих отраслях во многом совпадают	
Метод	Метод власти и подчинения	Метод координации

Рисунок 1.7 – Шаблон будущего слайда, содержащий таблицу

Слайд 10

ЗАГОЛОВОК. Автоцистерна пожарная АЦ 3,2 40 (4333)

Пожарные автоцистерны предназначены для тушения пожаров, доставки к месту вызова боевых расчетов, огнетушащих веществ и пожарно-технического вооружения.

Рисунок 1.8 – Шаблон будущего слайда, содержащий рисунок

И последняя возможная ситуация – это когда шаблон содержит математические формулы (рисунок 1.9). Также следует учесть, что формулы

могут быть набраны как с помощью редактора Microsoft equation 3.0, так и с помощью более новой версии редактора формулы – MathType 6.0 equation. Работы с разными версиями редакторов программно заметно отличается, что также важно учитывать для корректного оформления слайда.

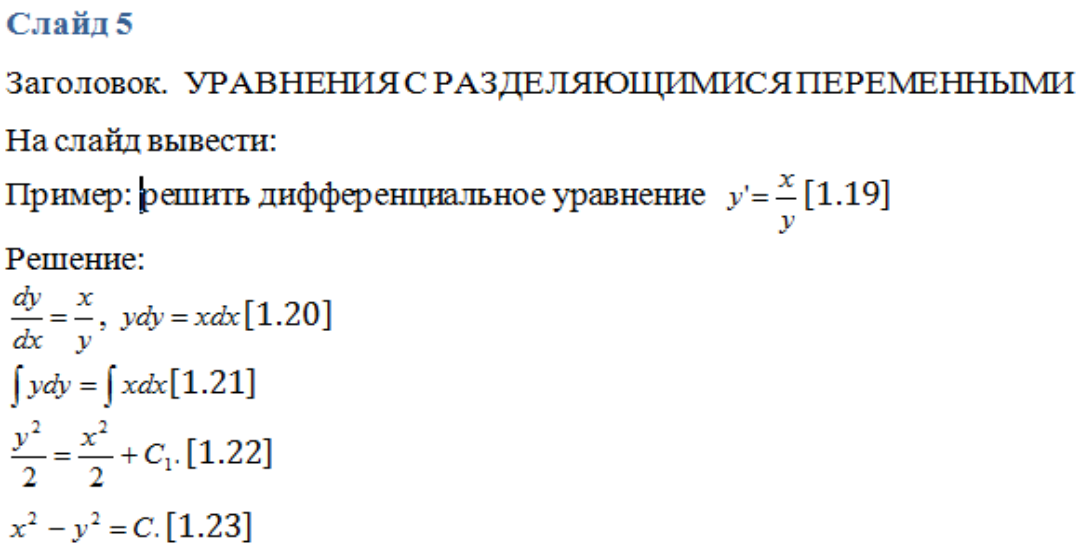

Рисунок 1.9 – Шаблон будущего слайда, содержащий математические формулы

Анализ входных данных позволяет понять, по какому принципу создаются текстовые документы. Далее необходимо описать данные в виде пригодном для обработки на компьютере.

1.5 Формализация исходных данных

Так как документы на основе, которых создаются презентации содержат различные объекты, то для того чтобы изучить свойства и выделить основные характеристики объектов необходимо провести формализацию данных. Формализация – процесс построения информационных моделей с помощью формальных языков [5].

Для формализованного описания объектов документа была использована реляционная модель данных. Данная модель ориентирована на организацию данных в виде отношений (таблиц). Для манипулирования

данными в реляционной модели используются: реляционная алгебра и реляционное исчисление [4].

Для шаблона на рисунке 1.6 можно выделить два множества объектов: таблица и абзацы. Отношение в реляционной модели содержит информацию об объектах. Отношение можно представить в виде таблицы. В данном случае отношение «абзацы» содержит информацию, которую необходимо знать при переносе текста на слайд, а так же при его последующем форматировании. К этой информации, относится: каким по счету абзац идет в шаблоне, является ли абзац заголовком, является ли абзац списком и содержит ли абзац слова выделенные «жирным».

Пусть дано множество доменов $D_1 = 1, 2, 3, 4$, $D_2 = да, нет, нет, нет$, $D_3 = нет, нет, нет, нет,$, $D_4 = да, нет, да, да$, тогда отношение R («Абзацы») над множеством доменов называется множеством упорядоченных кортежей (строк таблицы) вида $< t_1, t_2, \dots t_n >$, где $t_i \in D_i$. Пример кортежа в данном случае: $t_1 = 1, да, нет, да$. Таким образом, отношение R является подмножество, декартового произведение $R = D_1 \times D_2 \times D_3 \times D_4$ (таблица 1.1).

Таблица 1.1 – Отношение “Абзацы”

Абзацы			
Номер абзаца	Заголовок	Маркированный список	Текст выделенный, жирным
1	да	нет	да
2	нет	нет	нет
3	нет	нет	да
4	нет	нет	да

Как уже говорилось для того чтобы манипулировать данными отношений, используется реляционная алгебра и реляционное исчисление. Реляционное исчисление основано на логике первого порядка (исчисления предикатов). Выражение реляционного исчисления записывается в виде (формула (1)):

$$t \quad \varphi(t) \ , \tag{1}$$

где t – свободная переменная (кортеж),

$\varphi(t)$ – некоторая формула.

Для ситуации, когда необходимо определить является ли абзац заголовком можно использовать операцию проекция.

Далее атрибуты отношения R, обозначаются как: a – столбец «Номер таблицы», b – столбец «Заголовок», c – столбец «Маркированный список» и d – столбец – текст выделенный «жирным». В реляционном исчисление кортежей, запрос (операция проекции), записывается в следующем виде (формула (2)):

$$R.a, R.b \quad R \ , \tag{2}$$

где R – отношение («Абзацы», таблица 1.1),

R.a – указывает на атрибут отношения R.

В реляционной алгебре операция проекция определена следующим образом: $\pi_{a,b} \ R$. Результатом операции проекция является таблица (отношение) 1.2.

Таблицы 1.2 – Результат операции проекция

Номер абзаца	Заголовок
1	Да
2	Нет
3	Нет
4	Нет

Следующий запрос извлекает данные для случая, когда заголовок перенесен на слайд. Данный запрос в реляционном исчислении записывается в следующем виде (формула (3)):

$$r.a\ r.b\ r.c\ r.d \quad R \qquad (r.a > 1 \wedge r.b = \text{"нет"}) \ , \tag{3}$$

где $r.a$ –представляет атрибут (r кортеж отношения R).

В формуле (3) задается условия – номер абзаца должен быть больше единицы и абзац не должен представлять заголовок слайда. В результате получено отношение состоящие из всех кортежей отношения, за исключением первого кортежа. Аналогичная операция в реляционной алгебре $\sigma_{(a > 1 \quad b = \text{"нет"})} R$.

Для шаблона на рисунке 1.9 можно определить два отношения. Первое описывает – номер абзаца, количество формул в абзаце, в каком редакторе формул набран (если в абзаце присутствуют формулы). Второе отношение – номер абзаца, является ли абзац заголовком. Первое отношение Q = {(1, 2, 3, 4, 5, 6, 7, 8), (0, 0, 1, 0, 3, 2, 2), (Microsoft equation 3.0, Microsoft equation 3.0, Math type 6.0, Microsoft equation 3.0 , Math type 6.0, Microsoft equation 3.0, Math type 6.0)}. Второе отношение Z = {(1, 2, 3, 4, 5, 6, 7, 8), (да, нет, нет, нет, нет, нет, нет, нет)}. Используя отношения Q и Z можно получить, отношение которое, например, будет содержать информацию о тех абзацах, которые содержат формулы, при этом формулы должны быть набраны в определенном редакторе. Пусть Q = (a, b, c), где A, B и C представляют соответствующие атрибуты, аналогично с Z = (a, b). Таким образом, запрос в реляционном исчислении (формула (4)):

$$q.a \quad Q \quad \exists z: Z(z) \quad (\ q.c = \text{"}Math\ type\ 6.0\text{"} \wedge q.b > 0 \wedge z.b = \text{"нет"}\ ) \ , \tag{4}$$

где q – кортеж отношения Q,

z – кортеж отношения Z.

В формуле (4) так же использовался квантор существования (∃). Квантор существования, использовался, потому что среди свободных кортежей (справа от черты в формуле) нет кортежа, принадлежащего

отношению Q. Квантор существования показывает, что запрос должен вернуть атрибут, содержащий, по крайней мере, один кортеж. Аналогичный запрос в реляционной алгебре: $\pi_{Q.a}\,\sigma_{(Q.c=\text{"Math type 6.0"}\,\wedge\,Q.b>0\,\wedge\,Z.b=\text{"нет"}}\,Q\,\times \mathrm{Z}$. Результатом запроса является отношение состоящие из одного атрибута $W = \{(3,\ 5,\ 6)\}$, содержащего позиции параграфов в тексте, которые удовлетворяют условию запроса.

Вывод по первой главе

В данной главе было проанализировано предприятие, чья деятельность является объектом автоматизации. Также были проанализированы и формализованы исходные данные и рассмотрены требования к создаваемым презентациям. В результате проделанной работы можно определиться со средствами, которые необходимы для достижения поставленной цели, а также сформировать требования к приложению и разработать его структуру.

Глава 2. Проектирование приложения для создания презентаций на основе текстовых документов

2.1 Выбор средств реализации

При выборе языка программирования были рассмотрены следующие языки:

- Java;
- C#;
- C++;

Java представляет собой строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Все программы на языке Java транслируются в байт-код с помощью виртуальной машины JVM, что делает приложения Java платформо-независимыми [15]. Для взаимодействия с документами Microsoft Office можно использовать библиотеку Apache POI [12].

C++ – язык программирования общего назначения, он поддерживает разнообразные парадигмы программирования (например, объектно-ориентированное программирование). Основная область применения данного языка – это системное программирование. Однако C++ успешно используется во многих областях разработки приложений. Взаимодействие с офисными документами осуществляется посредством использования COM объектом.

C# – объективно-ориентированный язык программирования, предназначенный для разработки приложений, выполняющихся в среде .NET Framework. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений [14]. Для взаимодействия с приложениями Microsoft Office из .NET можно использовать технологию COM Interop или комплект разработки Open XML

SDK. Так же существует большое количество библиотек от сторонних разработчиков.

В качестве языка программирования был выбран язык C#, так как разработанное программное обеспечение будет функционировать под операционной системой Windows, также для языка программирования C# существует достаточно большое количество средств, с помощью которых можно взаимодействовать с документами Microsoft Office, что существенно упростит создание приложения.

В качестве среды разработки была выбрана Microsoft Visual Studio. Данная среда имеет большой набор сервисов, поддерживающие плагины от сторонних разработчиков, что позволяет расширять возможности среды разработки, предоставляет широкий набор функций для разработки под операционной системой Windows. Также Visual studio поддерживает технологию Windows Form. Windows Form предоставляет API для программирования графических приложений. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код – классы, реализующие API для Windows Forms, не зависят от языка разработки [8].

2.2 Анализ способов взаимодействия с документами

Ниже рассматриваются средства, которые позволяют взаимодействовать приложениям, написанным на языке C# с приложениями Microsoft Office. Рассматривались те средства, которые позволяют создавать презентации из текстовых документов и предоставляют необходимые функции для обработки презентаций. Кроме рассмотренных технологий (библиотек) существует достаточно большое количество различных средств. Например, DocX или NPOI для работы с текстовыми документами. Однако такие средства либо не предоставляют нужных функций, либо не способны осуществить создание презентации на основе текстового документа.

2.2.1 Open XML SDK

Начиная с версии Microsoft Office 2007, используется новый стандарт разметки Open XML для офисных документов. Переход на новый формат файлов вызван потребностью преодолеть ограничения, которые накладывали традиционные двоичные форматы хранения документов. Новый формат, базирующийся на формате XML, стандартизирован и основан на открытых технологиях, что обеспечивает его применимость на различных платформах и во многих ОС. Помимо переносимости, файлы, использующие XML формат в отличие от бинарного формата, позволяют просматривать содержимое, даже если часть документа повреждена. Стандарт включает три основных языка разметки. Для текстовых документов (Microsoft Word) используется WordprocessingML, для электронных таблиц (Microsoft Excel) SpreadsheetML и PresentationML — для презентаций (Power Point). Также существуют вспомогательные языки, например, DrawingML – он используется для работы с изображениями, схемами и т.п. Документы, созданные с помощью Open XML формата, представляет собой контейнер, состоящий из нескольких компонентов. Контейнером является сжатая ZIP-папка. Использования сжатия позволяет уменьшить размер файла. Новый стандарт кроме разметки документа также определяет структуру контейнера, и раздел спецификации, где описаны требования к совместимости кода разметки. Стандарт описывается в Open Packaging Convention (Открытом соглашении по правилам упаковки) [3].

Документы, созданные в Microsoft Word, имеют расширение docx. Данный формат, как и все форматы, основанные на Open XML, представляет собой ZIP-архив, который содержит составные части документа.

Пакет Open XML SDK позволяет работать с документами, которые соответствуют стандарту Open XML. Open XML SDK предоставляет типизированные классы для управления документами [13].

Основные компоненты Open XML представлены на рисунке 2.1.

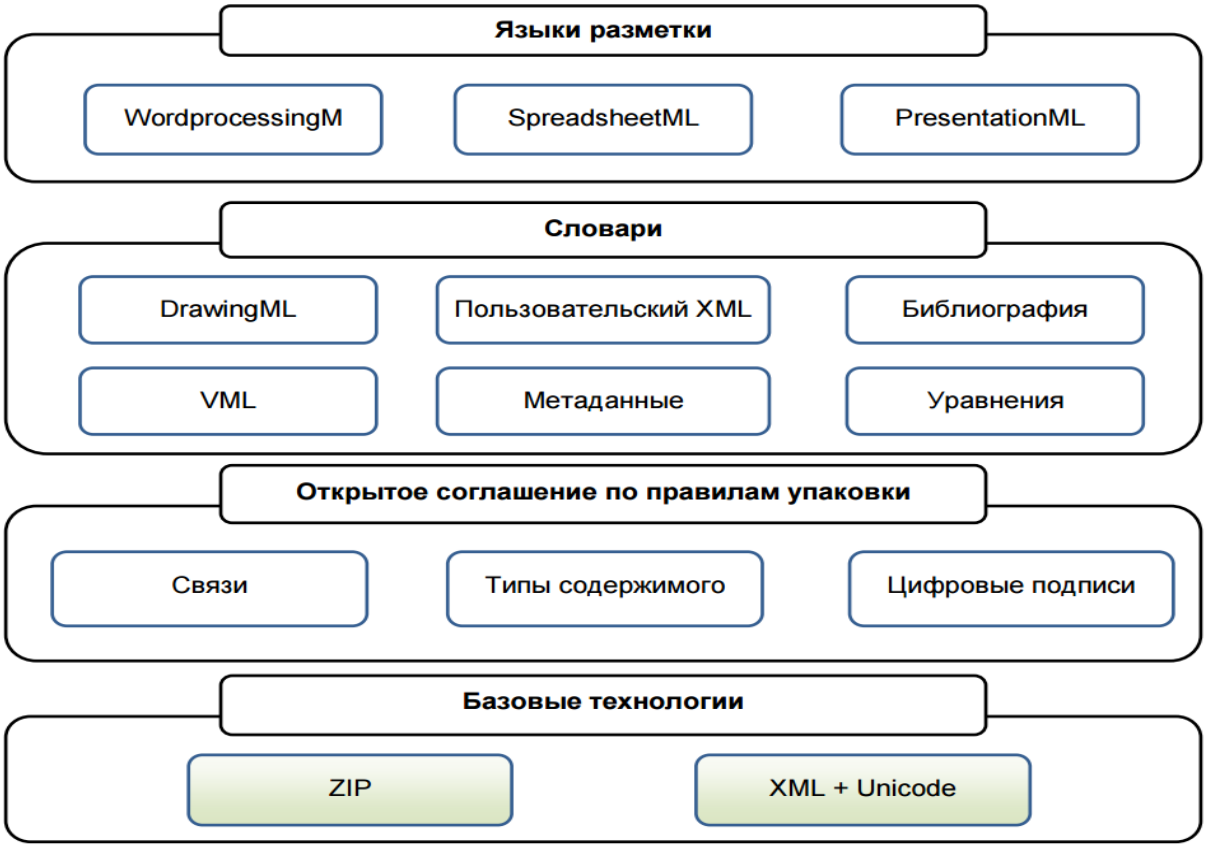

Рисунок 2.1 – Основные компоненты Open XML

Для работы с документами с помощью Open XML SDK не требуется установленных приложения Microsoft Office, но обработка документов с использованием данного пакета подразумевает работу с XML-деревом, что приводит к сложности реализации и читаемости кода.

2.2.2 COM Interop

COM Interop – технология, включённая в .NET CLR, позволяет объектам COM взаимодействовать с объектами .NET и наоборот. Данная технология обеспечивает доступ к существующим компонентам COM без необходимости модификации оригинальных компонентов [22]. Объектная модель Word и PowerPoint определена в пространствах имен Microsoft.Office.Interop.Word и Microsoft.Office.Interop.PowerPoint соответственно. Объектная модель предоставляет большое число различных объектов, с которыми можно взаимодействовать. Эти объекты организованы

в виде иерархии, которая точно соответствует пользовательскому интерфейсу. Таким образом, работа с COM объектами офисных приложений значительно упрощает обработку документов. Ключевым объектом в объектной модели Word является объект Application. Данный объект представляет запущенное приложение Microsoft Office, Application включает в себя все остальные объекты. Как только запускается Word, создаётся коллекция Documents. При запуске приложения создается коллекция Documents, она содержит как минимум один документ (объект Document). Document в свою очередь содержит метод Range, возвращающий объект Range. Данный объект позволяет работать с выбранным диапазоном. Из выбранной области с помощью методов класса Range можно получить доступ к абзацу, предложению, таблице и т.д. [1, 9]. Объектная модель Word представлена на рисунке 2.2.

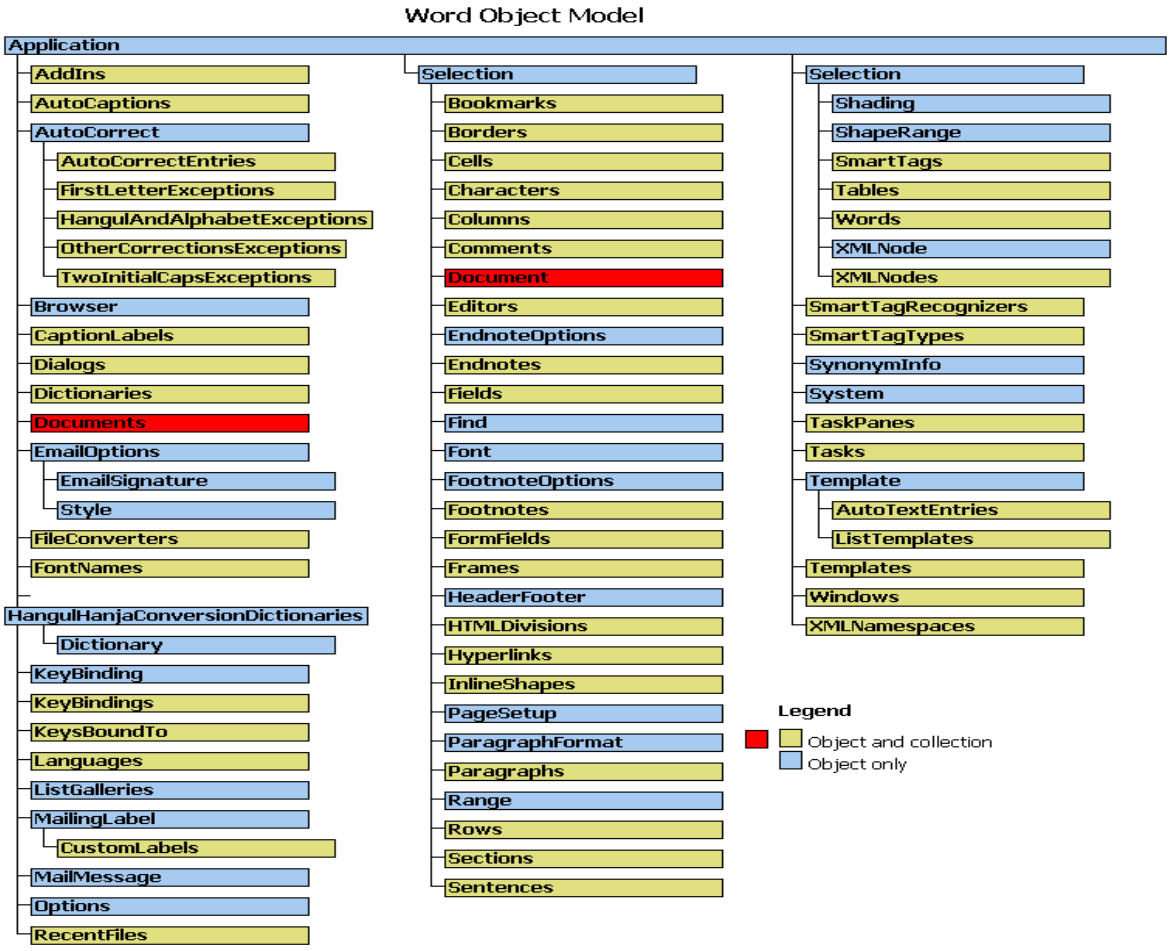

Рисунок 2.2 – Объектная модель Word

Объектная модель PowerPoint во многом схожа с моделью Word. COM Intrerop является мощным и простым средством для работы с документами, но для работы с COM необходимы установленные приложения Microsoft Office.

2.2.3 Aspose API

Aspose, поставщик API форматов файлов для .NET, Java, Android, SharePoint, Reporting Services для преобразования и автоматизации обработки документов. Он работает с документами, созданными в Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Portable Document Format и OpenDocument. Работа с API Aspose проста и не требует установленных приложений Microsoft Office, однако средства Aspose являются платными.

Для работы с текстовыми документами используется Aspose.Words. Aspose.Words для .NET - это усовершенствованный API обработки документов Word, который позволяет выполнять широкий спектр задач непосредственно в .NET-приложениях. Aspose.Words позволяет выполнить все типичные задачи: обработка текстов, создание таблиц и так далее. Aspose.Words не требует установленного приложения Microsoft Word [17].

Aspose.Slides – API для управления презентациями. Этот API-интерфейс позволяет любому .NET-приложению читать, записывать, изменять и преобразовывать документы PowerPoint. Aspose.Slides позволяет, работать с текстом, рисунками, таблицами, а так же с другими объектами [16].

2.2.4 Выбор средства взаимодействия с документами

Исходя из приведенного в прошлом параграфе описания можно, выделить основные плюсы и минусы, рассмотренных технологий для взаимодействия с документами.

В таблице 2.1 представлен сравнительный анализ способов взаимодействия с документами.

Таблица 2.1 – сравнительная характеристика

Характеристики	Open XML SDK	COM Interop	Aspose API
Простота использования	-	+	+
Бесплатность	+	+	-
Требуемая функциональность	+	+	+
Итог	2	3	2

Таким образом, исходя из анализа для взаимодействия с документами, была выбрана технология COM Interop. При анализе не была рассмотрена такая характеристика технологий (библиотек), как требования наличия установленных приложений Microsoft Office. Данная характеристика не была рассмотрена, так как в отделе сопровождения дистанционного обучения для работы с текстовыми документами и презентациями при создании учебного материала основными используемыми программными продуктами являются: Microsoft Word версии 2007, а также Microsoft PowerPoint версии 2007. Поэтому, в данном случае требование к наличию установленных приложений Microsoft Office не является принципиальным.

2.3 Формирование требований к программному обеспечению

Выработка требований – это деятельность, направленная на выявлении, документирование, а также обслуживание требований, предъявляемых к разрабатываемому программному обеспечению. Данный этап разработки является наиболее важным, так как конечный продукт основывается на наборе требований [20].

Для моделирования основных функциональных требований к разрабатываемому программному обеспечению использовалась диаграмма вариантов использования.

Был выделен один актер: сотрудник отдела сопровождения дистанционного обеспечения. Так же были определенны два прецендента. Конвертация – данная функция осуществляет создание презентации на основе текстового документа. Вторая функция – настройки оформления презентации (настройка шрифтов, темы оформления и т.д).

В таблицах 2.2 и 2.3 представлена спецификация прецендентов.

Таблица 2.2 – Описание прецедента “Конвертация документов”

Прецендент: Конвертация документов
ID: 1
Краткое описание: Приложение создает презентацию на основе текстового документа.
Главные актеры: Сотрудник
Второстепенные актеры: нет
Предусловие 1. Сотрудник запустил программу.
Основной поток: 1. Сотрудник выбирает документ. 2. Приложение получает путь к документу. 3. Сотрудник указывает путь, по которому будет сохранена презентация. 4. Сотрудник запускает обработку. 5. Пока пути к документам недействительны 5.1 Приложение просить правильно указать путь 6. Приложение создает презентацию.
Постусловие: Приложение создало презентацию на основе текстового документа.

Продолжение таблицы 2.2

Альтернативные потоки: нет

Таблица 2.3 – Описание прецендента “Настройки оформления”

Прецендент: Настройки оформления
ID: 2
Краткое описание: Пользователь выбирает настройки оформления презентации.
Главные актеры: Сотрудник
Второстепенные актеры: нет
Предусловие 1. Сотрудник открывает окно настройки.
Основной поток 1. Сотрудник выбирает размер шрифт и размер шрифта текста. 2. Сотрудник выбирает тему оформления. 3. Сотрудник выбирает цвет текста.
Постусловие: Приложение получило настройки выбранные сотрудником.
Альтернативные потоки: нет

На рисунке 2.3 представлена диаграмма вариантов использования.

Разработанная диаграмма помогает выделить основные функции и определить то, как должны вести себя компоненты приложения для того чтобы реализовать нужную для пользователя функциональность.

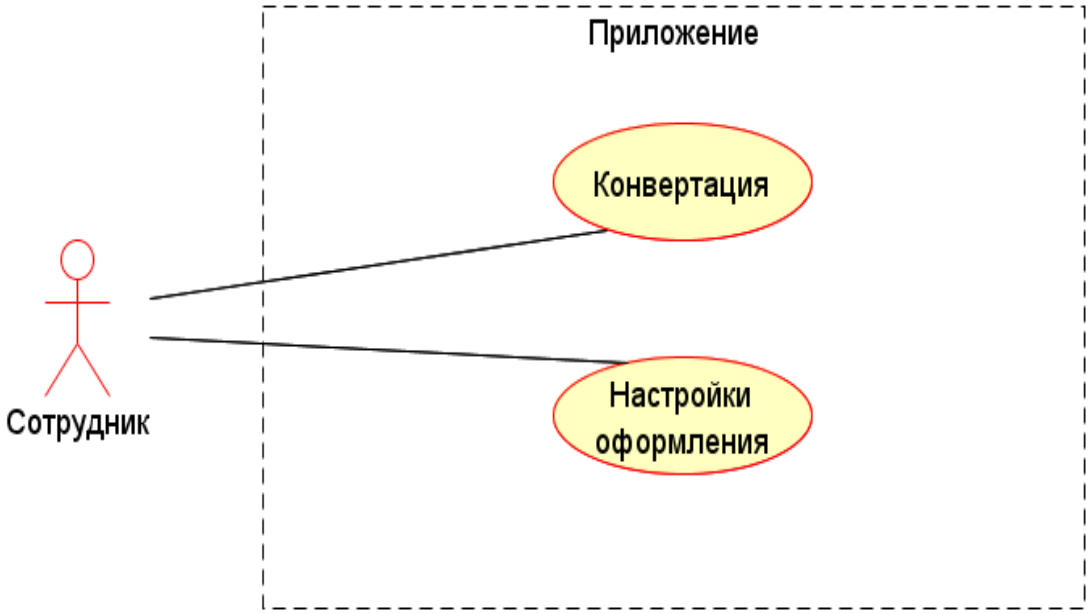

Рисунок 2.3 – Диаграмма вариантов использования

Далее необходимо разработать структуру приложения. Для моделирования структуры использовалась диаграмма классов UML.

2.4 Диаграмма классов разрабатываемого приложения

Диаграмма классов для разрабатываемого приложения представлена на рисунке 2.4.

Класс DocWord предназначен для работы с Microsoft Word. В данном классе открывается нужный документ. Так же в DocWord осуществляется перебор всех шаблонов слайдов.

Класс PowerPoint предназначен для работы с презентацией. В этом классе создается новая презентация, также определяется тип объекта (таблица, рисунок и т.д), который должен быть перенесен на слайд.

ProcessignText – класс для работы с текстом презентации.

ProcessingImage – класс для обработки рисунков.

ProcessingMath – класс для обработки математических формул.

Класс ProcessingTable – класс для работы с таблицами.

Класс SettingPresentationDefault – содержит настройки презентации по-умолчанию.

Класса SettingPresentation – содержит настройки измененные пользователем.

Также были определены отношения между классами.

Между экземплярами классов DocWord и PowerPoint определено отношение – композиция. Так как объект класса PowerPoint является составной частью DocWord, а также объект класса DocWord управляет “жизнью” объекта класса PowerPoint.

Между экземплярами классов PowerPoint и ProcessignText, ProcessingImage, ProcessingMath, ProcessingTable также определенно отношение композиция.

Между экземплярами классов SettingPresentation SettingPresentationDefault установлено отношение обобщение (наследование).

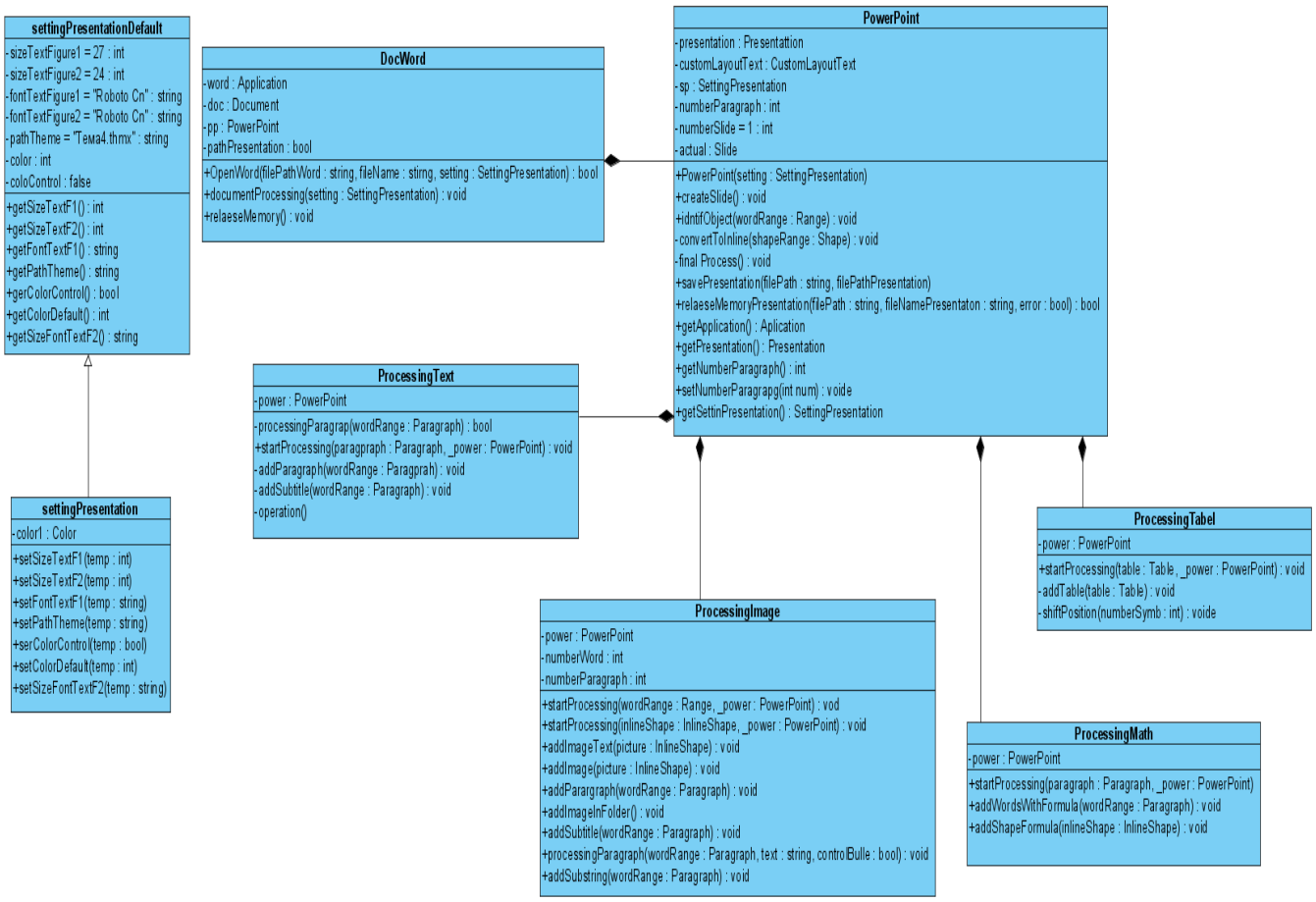

Рисунок 2.4 – Диаграмма классов

Диаграмма классов является одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. Диаграмма классов не отображает динамическое поведение объектов изображенных на ней классов [19].

Вывод по второй главе

Во второй главе были выбраны средства реализации (язык программирования, среда разработки), средства для программной работы с документами, сформированы основные требования к приложению, а также разработана его классовая структура. Таким образом, определившись со всеми средствами и набором функций, необходимо перейти к реализации приложения.

Глава 3. Реализация приложения для создания презентаций на основе текстовых документов

3.1 Реализация приложения

Так как для работы с документами в отделе сопровождения дистанционного обучения, используют пакет приложений Microsoft Office 2007, то приложение было реализовано для работы именно с Microsof Office 2007. Так же приложения, написанные для работы с более старой версией Office, будут совместимы и с более новыми версиями. Но если наоборот приложение написано для более новой версией, то при работе со старой версией возможны проблемы совместимости. Весь текст, таблицы и рисунки оформлялись в соответствии с требованиями, описанными в параграфе 1.3 Главы 1.

Первым был реализован класс DocWord. Данный класс является главным классом приложения. В методе OpenWord класса DocWord (рисунок 3.1) сначала создается копия документа, который является основой для презентации. Копия создается для того, чтобы не повредить данные в исходном документ. Далее создаётся объект Application, вследствие чего запускается приложение Microsoft Word. После завершения обработки документа копия будет удалена. Приложение запускается в фоновом режиме для того чтобы пользователь случайно не прервал обработку. После запуска приложения открывается копия нужного текстового документа.

```
public void OpenWord(string filePathWord, Form1 temp, string fileName, SettingPresentation setting)
{
      try
       {
       System.IO.FileInfo pathWord = new System.IO.FileInfo(filePathWord);
       System.IO.FileInfo copyPathWord = pathWord.CopyTo(System.IO.Path.GetFileNameWithoutExtension(filePathWord), true);
       Forma = temp;
       fileNamePresentation = fileName;
       Object confirmConversions = Type.Missing;
       Object readOnly = false;
       word = new WordInterop.Application();
       word.Visible = false;
       doc = new Document();
       doc = word.Documents.Open(copyPathWord.FullName, ref confirmConversions, ref readOnly); // открытие документа
       documetnProcessing(setting); // запуск обработки документа
       copyPathWord.Delete();
       }
       catch(Exception)
       {
         relaeseMemory();
       }
}
```

Рисунок 3.1 – Метод OpenWord

После открытия документа вызывается метод documentProcessing. В данном методе создается объект Range и экземпляр класса PowerPoint.

В этом методе орагинзуется цикл для перебора всех шаблонов слайдов (рисунок 3.2). Перебор осуществляется следующим образом. С помощью метода Find класса Range выполняется поиск заголовка (номер слайда). Заголовок “сообщает” о том, что должен быть создан новый слайд. Данные под слайдом выделяются (метод SetRange), после чего методу класса PowerPoint identifObject передается объект Range.

```
while (flag)
{
    flag = wordRange.Find.Execute("Слайд " + i);
    start = wordRange.End;
    wordRange.Start = wordRange.End;
    flag = wordRange.Find.Execute("Слайд " + (i + 1));
    if (!flag)
        end = doc.Content.End;
    else
        end = wordRange.Start;
    wordRange.SetRange(start, end);
    wordRange.Select();
    pp.identifObject(wordRange);
    i++;
}
```

Рисунок 3.2 – Цикл для перебора шаблонов слайдов

После обработки необходимо закрыть документ и само приложение Microsof Word. Если приложение не закрыть, то оно так и останется в памяти.

В случае если вовремя обработки документа произойдёт ошибка (например документ будет удален) и будет вызвано исключение, то исключение будет перехвачено (рисунок 3.1) и приложение будет закрыто.

Для работы с PowerPoint необходимо учесть две важные особенности.

Таблицы, математические формулы и рисунки можно перенести из документа Word на слайды PowerPoint только путем их копирования через буфер обмена. Для того чтобы вставить объект из буфера обмена на слайд,

приложение PowerPoint должно быть запущено не в фоновом режиме. Данная особенность характерна для PowerPoint версий 2007-2013, и только в версии Microsoft Office 2016 можно осуществить копирование объектов при скрытом приложении.

Вторая особенность, которую необходимо учесть, относится к правильному освобождению памяти. Так же как при работе с Word посредством COM-объектов, так и при работе с PowerPoint создается процесс приложения. Но в отличие от Word, где для того чтобы освободить память нужно закрыть документ и приложение, при работе с PowerPoint этого будет недостаточно. Сложность с освобождением памяти возникает из-за того, что COM-объекты выполняется не под управлением CLR (COM-объекты являются неуправляемым кодом). Для того чтобы работать с COM в CLR используется посредник называемый – Runtime Callable Wrapper (RCW). Хотя оболочка RCW выглядит для приложений на .NET обычным объектом, ее основная функция состоит в маршалинге (маршалинг нужен для передачи данных между управляемым и неуправляемым кодом) вызовов между клиентом .NET и COM-объектом. RCW реализует интерфейсы, реализуемые COM-объектом, и предоставляет доступ к методам, свойствам и событиям из интерфейсов объекта. То есть, работая с COM в .NET, работа на самом деле ведется не с самим COM-объектом, а с оболочкой RCW представляющей конкретный COM-объект. COM-объект содержит счетчик ссылок, если счетчик больше нуля значит объект используется и он остается в памяти, как только счетчик становится равным нулю объект уничтожается. При работе с COM в .NET счетчик ссылок имеется у оболочки RCW, которая представляет COM-объект. Чтобы сбросить счетчик ссылок до нуля используется метод ReleaseComObject класса Marshal. Таким образом, передав методу ReleaseComObject COM-объекты PowerPoint, счетчики COM-объектов будут обнулены, и приложение PowerPoint будет закрыто, а память освобождена [6].

Для работы с презентациями был реализован класс PowerPoint. Экземпляр класса PowerPoint создается в методе documentProcessing класса DocWord. В констукторе класса PowerPoint запускается приложение PowerPoint и создается презентация. К презентации применяется тема оформления, задаются размеры слайда и его положение (рисунок 3.3).

```
public PowerPoint()
{
    try
    {

        Point = new PP.Application(); //  power point
        Point.Visible = Office.msoTrue;
        presentation = Point.Presentations.Add(WithWindow: Office.msoTrue); //Создание презентации
        Point.WindowState = PpWindowState.ppWindowMinimized;
        presentation.ApplyTheme(sp.getPathTheme());// применение темы оформления
        presentation.PageSetup.SlideHeight = 33.867f * 28.35f;
        presentation.PageSetup.SlideWidth = 19.05f * 28.35f;
        presentation.PageSetup.SlideOrientation = MsoOrientation.msoOrientationHorizontal;
        customLayoutText = presentation.SlideMaster.CustomLayouts[PpSlideLayout.ppLayoutText];
        customLayoutText.Shapes[1].TextFrame2.TextRange.Font.Fill.ForeColor.RGB = Color.FromArgb(200, 143, 0).ToArgb();
    }
    catch(Exception)
    {
        MemoryRelaesePresentation("","",false);
    }
}
```

Рисунок 3.3 – Конструктор класса PowerPoint

Так же, класс PowerPoint содержит метод createSlide в котором осуществляется создание слайда. Основным методом класса является identifObject, он принимает объект Range. Далее можно получить доспут к абзацам. Затем в цикле осуществляется перебор коллекции, для того чтобы определить какой объект (текст, таблица и так далее) содержит Paragraph (объект коллекции Paragraphs, представляющий абзац). В зависимости от выделенного объекта создается экзепмляр нужного класса (например класс для обработки таблиц). Фрагмент метода identifObject представлен на рисунке 3.4. Особое внимание при определении объекта находящегося в Paragraph следует уделить рисункам. Рисунки, которые доступны для копирования на слайд являются членами коллекции InlineShapes. InlineShape – представляет собой рисунок у которого обтекание задано – в тексте. Shape – рисунок у которого обтекание не в тексте, чтобы рисунок представляемый

объектом Shape можно было скопировать его необходимо конвертировать в InlineShapes с помощью соответсвующего метода. Так же в InlineShape может, находится формула, набранная с помощью редактора формул Microsoft equation 3.0. И при попытке скопировать формулу на слайд как рисунок, будет вызвано исключение. Чтобы избежать таких ситуаций необходимо, дополнительно проверять (рисунок 3.5), что именно (рисунок или формулу) представляет объект InlineShape.

```
foreach (WordInterop.Paragraph paragr in wordRange.Paragraphs)
{
    flag = false;
    if (paragr.Range.OMaths.Count > 0 || paragr.Range.InlineShapes.Count > 0)
    {
        flag = true;
        ProcessingMath procMath = new ProcessingMath();
        procMath.starProcessing(paragr, this);
    }
    if (paragr.Range.Tables.Count > 0)
    {
        flag = true;
        if (numberTable <= countTable)
        {
            ProcessingTable procTable = new ProcessingTable();
            procTable.startProcessing(paragr.Range.Tables[numberTable], this);
            numberTable++;
        }
    }
    if ((paragr.Range.InlineShapes.Count > 0 || paragr.Range.ShapeRange.Count > 0) && wordRange.OMaths.Count < 1)
    {
        if (paragr.Range.ShapeRange.Count > 0)
        {
            int count = paragr.Range.ShapeRange.Count;
            for (int i = 1; i <= count; i++)
                convertToInline(paragr.Range.ShapeRange[i]);
        }
        for(int i=1; i <= paragr.Range.InlineShapes.Count; i++)
        {
            bool inlineShape;
            bool oleObject;
            bool smart;
            bool diagram;
            inlineShape = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapePicture;
            oleObject = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapeEmbeddedOLEObject;
            smart = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapeSmartArt;
            diagram = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapeDiagram;
            if (inlineShape || oleObject || smart || diagram)
            {
                flag = true;
                bool equation = (oleObject ? paragr.Range.InlineShapes[i].OLEFormat.ClassType != "Equation.3" : false);
                if (equation)
                {
                    ProcessingImage procImage = new ProcessingImage();
                    procImage.startProcessing(paragr.Range.InlineShapes[i],this);
                    break;
                }
            }
        }
    }
```

Рисунок 3.4 – Фрагмент метода identifObject

```
if (wordRange.InlineShapes[1].Type == WdInlineShapeType.wdInlineShapeEmbeddedOLEObject)
{
    if (wordRange.InlineShapes[1].OLEFormat.ClassType != "Equation.3")
        controlEquaction = true;
}
```

Рисунок 3.5 – Проверка объкта InlineShape

После того как все члены коллекции Paragraphs были индентицированны (рисунок 3.4) и перенесенны на слайд, вызывается метод finalProcess, в котором осуществляется проверка, того присутсвует ли таблиц на слайде, если да и на слайде нет текста, то таблица будет выровнена по центру. Затем если требуется, высота таблицы увеличивается и далее её содержимое форматируется в соотвествии с требованиями. После окончания обработки презентация сохраняется в укзанную директорию, осовобождаются COM-объекты PowerPoint и приложение PowerPoint закрывается. В случае возникновения исключительных ситуаций, исключения будут перехвачены обработчиком исключений, вся память будет освобождена и приложение PowerPoint будет закрыто.

Если в объекте Paragraph находится текст, то создается экземпляр класса ProcessingText. В данном классе проверяется, является ли текст в Paragpraph заголовком (слово заголовок перед текстом), если да то текст помещается в фигуру слайда, представляющую заголовок и обрабатывается в соответствии с требованиями. Фрагмент кода для обработки заголовка представлен на рисунке 3.6.

```
if (controlHeader)
{
    header = wordRange.Range.Text;
    power.actual1.Shapes[1].TextFrame2.TextRange.Text = header.Substring(10).Trim();
    int lines = power.actual1.Shapes[1].TextFrame2.TextRange.Lines.Count;
    if (lines > 1)
        power.actual1.Shapes[1].TextFrame2.TextRange.InsertBefore("\r");
    bool comprStr = header.IndexOf("Раздел", StringComparison.OrdinalIgnoreCase) != -1;
    if (!power.sp1.getColorControl)
    {
        int i = 2;
        power.actual1.Shapes[1].TextFrame2.TextRange.Words[1].Font.Fill.ForeColor.RGB = power.sp1.getColorDefault;
        while (comprStr && 4 > i)
        {
            string temp = power.actual1.Shapes[1].TextFrame2.TextRange.Words[i].Text;
            power.actual1.Shapes[1].TextFrame2.TextRange.Words[i].Font.Fill.ForeColor.RGB = power.sp1.getColorDefault;
            if (temp.Contains("."))
                break;
            i++;
        }
    }
    else
    {
        int i = 2;
        power.actual1.Shapes[1].TextFrame2.TextRange.Words[1].Font.Fill.ForeColor.RGB =
            Color.FromArgb(1, power.sp1.setColor1.B, power.sp1.setColor1.G, power.sp1.setColor1.R).ToArgb();
        int length = power.actual1.Shapes[1].TextFrame2.TextRange.Words.Count;
        while (comprStr && 4 > i)
        {
            string temp = power.actual1.Shapes[1].TextFrame2.TextRange.Words[i].Text;
            power.actual1.Shapes[1].TextFrame2.TextRange.Words[i].Font.Fill.ForeColor.RGB =
            Color.FromArgb(1, power.sp1.setColor1.B, power.sp1.setColor1.G, power.sp1.setColor1.R).ToArgb();
            if (temp.Contains("."))
                break;
            i++;
        }
    }
}
```

Рисунок 3.6 – Фрагмент кода для обработки заголовка

Кроме так заголовка так же может быть подзаголовок, в этом случае создается ещё одна фигура и в неё помещается текст, который форматируется, так же как и текст заголовка.

Если в Paragraph находится обычный текст, то он помещается в основную фигуру. Далее текст так же форматируется. Фрагмент кода для форматирования основного текста представлен на рисунке 3.7.

```
int numberWord = 1;
int amountWordsSlide = power.actual1.Shapes[2].TextFrame2.TextRange.Paragraphs[power.numberParagraph1].Words.Count;
int amountWord = wordRange.Range.Words.Count;
for (int i = 1; i <= amountWordsSlide; i++)
{
    if (numberWord <= amountWord)
    {
        int bold = wordRange.Range.Words[numberWord].Font.Bold;
        if (bold == -1)
        {
            power.actual1.Shapes[2].TextFrame2.TextRange.Paragraphs[power.numberParagraph1].Words[i].Font.Fill.ForeColor.RGB = Color.FromArgb(63, 63, 63).ToArgb();
            string wordActual = power.actual1.Shapes[2].TextFrame2.TextRange.Paragraphs[power.numberParagraph1].Words[i].Text.Trim();
            if ((wordActual.Contains("-") || wordActual.Contains(".")) && wordActual.Length > 1)
                numberWord++;
        }
    }
    numberWord++;
}
flag = true;
}
if (wordRange.Alignment == WdParagraphAlignment.wdAlignParagraphCenter)
    power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).ParagraphFormat.Alignment = PpParagraphAlignment.ppAlignCenter;
if (wordRange.Alignment == WdParagraphAlignment.wdAlignParagraphLeft)
    power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).ParagraphFormat.Alignment = PpParagraphAlignment.ppAlignLeft;
if (wordRange.Alignment == WdParagraphAlignment.wdAlignParagraphRight)
    power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).ParagraphFormat.Alignment = PpParagraphAlignment.ppAlignRight;
if (wordRange.Alignment == WdParagraphAlignment.wdAlignParagraphJustify)
    power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).ParagraphFormat.Alignment = PpParagraphAlignment.ppAlignJustify;
return flag;
```

Рисунок 3.7 – Фрагмент кода для форматирования основного текста

В приведенном на рисунке 3.7 коде каждое слово сравнивается с соответствующим словом из текстового документа, для того чтобы определить выделено ли слово курсивом (жирным или подчеркнуто) и если выделено, то соответствующе отформатировать. Далее текст, слайда выравнивается так же в соответствии с текстом из документа Word.

Для работы с таблицами используется класс ProcessingTable. Таблица копируется на слайд, через буфер обмена. Чтобы расположить таблицу в тексте, в основную текстовую фигуру слайда добавляется новый абзац и в новый абзац ставится метка (например, @). Затем координаты метки

присваиваются координатам таблицы и таким образом, таблица будет располагаться в тексте. Код метода shiftPosition представлен на рисунке 3.8.

```
private void shiftPosition(int numberSymb)
{
    PP.Shape shps = power.actual1.Shapes[power.actual1.Shapes.Count];
    if(shps.Type != MsoShapeType.msoTable)
    {
        for (int i = 1; i < power.actual1.Shapes.Count; i++)
        {
            shps = power.actual1.Shapes[i];
            if (shps.Type == MsoShapeType.msoTable)
                break;
        }
    }
    shps.Left = (power.presentation1.PageSetup.SlideWidth - shps.Width) / 2;
    shps.Top = power.actual1.Shapes[2].TextFrame.TextRange.Find("@").BoundTop + 5;
    power.actual1.Shapes[2].TextFrame.TextRange.Find("@").Delete();
    power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text = "\t\r";
    power.numberParagraph1++;
    while (true)
    {
        power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Words(numberSymb).Text = "@";
        float bountTop = power.actual1.Shapes[2].TextFrame.TextRange.Find("@").BoundTop;
        float topShape = shps.Top;
        float heigthShape = shps.Height;
        if (bountTop < topShape + heigthShape)
        {
            power.actual1.Shapes[2].TextFrame.TextRange.Find("@").Delete();
            power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text = "\t\r";
            power.numberParagraph1++;
        }
        else
        {
            power.actual1.Shapes[2].TextFrame.TextRange.Find("@").Delete();
            break;
        }
    }
}
```

Рисунок 3.8 – Метод shiftPosition

Корректно скопировать таблицу через буфер обмена можно, только в том случае, если содержимым таблицы является текст. Если таблица содержит, например, рисунок, то при копировании все её содержимое выйдет за пределы ячеек и будет располагаться в случайных позициях слайда. Решить эту проблему можно следующим способом. Скопировать таблицу без содержимого, а затем содержимое из каждой ячейки таблицы текстового документа, копировать в соответствующие ячейки таблицы, расположенной на слайде. Однако для этого необходимо перебрать все ячейки таблицы, которая находится в текстовом документе, а если ячейки объединены, то при

попытке получить к ним доступ будет вызываться исключение. В таком случае единственным решением данной проблемы является копирование таблицы как рисунка.

Для обработки рисунков используется класс ProcessingImage. Прежде чем создать экземпляр класса ProcessingImage проверяется количество рисунков и наличие текста в выделенной области. Если рисунков больше одного или в выделенной области нет текста, то вызывается перегруженный метод startProcessing, который принимает только рисунок, далее рисунок (рисунки) будет копироваться на слайд аналогично тому как было описано в пункте 3.1.4. для таблиц. То есть рисунок копируется на слайд, дальше ставится метка и координаты метки присваиваются координатам рисунка.

Если рисунок один и в выделенной области есть текст, то его необходимо поместить в фигуру “овал”. Чтобы добавить рисунок в фигуру необходимо указать для фигуры путь к рисунку. Есть два способа как получить путь к рисунку.

Первый способ – так как форма docx представляет собой архив, его можно разархивировать, переименовав например, в формат zip. В архиве в папке media находятся все рисунки, содержащиеся в документе. Затем можно получить путь к любому из рисунков. Но при таком подходе сложно сопоставить рисунок, который находится в выделенной области объекта Range, с рисунком находящимся в папке media.

Второй способ значительно проще. Нужный рисунок копируется через буфер обмена в корневую папку приложения с помощью класса Bitmap (рисунок 3.9), затем программа получает путь к рисунку и далее для фигуры указывается путь к данному рисунку.

```
var data = Clipboard.GetDataObject();
var image = (Image)data.GetData(DataFormats.Bitmap, true);
var currentBitmap = new Bitmap(image);
currentBitmap.Save(System.Windows.Forms.Application.StartupPath + @"\Image.png");
Clipboard.Clear();
```

Рисунок 3.9 – Копирование рисунка в корневую папку приложения

Овал на слайде располагается в правой части слайда. Возможны ситуации, когда текст будет “наползать” на овал, чтобы этого не произошло из каждого абзаца, извлекаются подстроки длиной в 46 символов. Далее текст обрабатывается, так же как было описано в пункте 3.1.3.

ProcessingMath – это класс для работы с формулами. Формулы могут быть набраны в редакторе формул Microsoft equation 3.0 или MathType 6.0 equation.

Если формула набрана в редакторе MathType 6.0 equation, то с такой формулой можно работать программно (создавать и изменять). Если формула набрана в Microsoft equation 3.0, то с такими формулами нет возможности работать программно и они (формулы) представляют собой объект InlineShape. В Microsoft PowerPoint 2007 формулы могут быть набраны только в Microsoft equation 3.0, поэтому в PowerPoint отсутствует возможность программно работать с формулами (начиная с версии Microsoft PowerPoint 2010, такая возможность появляется). Таким образом, формулы набранные, в MathType 6.0 equation будут изменяться (размер и цвет шрифта) в документе Word и только затем копироваться. А формулы, созданные, в редакторе Microsoft equation 3.0 будут копироваться, в том виде в каком они были созданы разработчиком учебного материала, так как в Microsoft PowerPoint 2007 нет возможности изменить формулы в соответствии с требованиями.

Перенос формул осуществляется следующим образом. Как и с таблицей, чтобы расположить формулы, в основной фигуре слайда содержащей текст, ставится метка. Но в отличие от таблиц для формул не создается нового абзаца, так как формулы могут располагаться в одной строке с текстом или с другими формулами. Код копирование и размещение представлен на рисунке 3.10. Чтобы текст не “наезжал” на формулы или формулы на формулы, проверяется, была ли вставлена формула в текущий абзац основной фигуры слайд. Если да, то перед меткой вставляются пробелы, до того момента пока позиция метки относительно левого края

слайда (Left), не станет больше позиции формулы относительно левого края слайда к которой прибавляется ширина формулы (Width).

```
private void addShapeFormula(WordInterop.InlineShape inlineShape)
{
    int numberSymb = power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text.Length;
    if (power.actual1.Shapes[power.actual1.Shapes.Count].Type == MsoShapeType.msoEmbeddedOLEObject)
    {
        if (power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text.Length > 0)
        {
            power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text += "\t";
            numberSymb = power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text.Length;
            power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Words(numberSymb).Text = "@";
        }
        else
            power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Words(numberSymb + 2).Text = "@";
    }
    else
        power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Words(numberSymb + 2).Text = "@";
    inlineShape.Range.Copy();
    power.application.ActiveWindow.View.PasteSpecial(PpPasteDataType.ppPasteOLEObject);
    power.actual1.Shapes[power.actual1.Shapes.Count].Top = power.actual1.Shapes[2].TextFrame.TextRange.Find("@").BoundTop;
    power.actual1.Shapes[power.actual1.Shapes.Count].Left = power.actual1.Shapes[2].TextFrame.TextRange.Find("@").BoundLeft;
    float leftShape = power.actual1.Shapes[power.actual1.Shapes.Count].Left;
    float topShape = power.actual1.Shapes[power.actual1.Shapes.Count].Width;
    float left = power.actual1.Shapes[2].TextFrame.TextRange.Find("@").BoundLeft;
    power.actual1.Shapes[2].TextFrame.TextRange.Find("@").Delete();
    int i = 1;
    while (left <= leftShape + topShape && i < 7)
    {
        power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text += "\t";
        numberSymb = power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Text.Length;
        power.actual1.Shapes[2].TextFrame.TextRange.Paragraphs(power.numberParagraph1).Words(numberSymb).Text = "@";
        left = power.actual1.Shapes[2].TextFrame.TextRange.Find("@").BoundLeft;
        power.actual1.Shapes[2].TextFrame.TextRange.Find("@").Delete();
        i++;
    }
}
}
```

Рисунок 3.10 – Метод addShapeFormula

Но так как Microsoft PowerPoint 2007 не позволяет программно работать с формулами, а именно изменять их размер то всё равно возможны ситуации, при которых формулы будут “наползать” на текст.

Для того чтобы пользователю было удобно работать с приложением был реализован графический интерфейс. Основное окно приложения представлено на рисунке 3.11.

Для того чтобы добавить путь к документу на основе которого будет создана презентация, пользователь должен нажать кнопку “Добавить документ”, при нажатии на кнопку открывается окно в котором пользователь указывает путь к нужному документу. Если при обработке документа произойдет ошибка, то для обработки будет запущен следующий документ из очереди. Так же можно добавить все файлы, из папки нажав на кнопку “Добавить папку”. При нажатии на кнопку “Удалить файлы” из очереди

будут удалены выделенные файлы. Кнопка “Конвертировать” запускает процесс создания презентации, так как данный процесс достаточно долгий, то для того чтобы предотвратить зависание приложения, преобразование документа осуществляется в отдельном потоке.

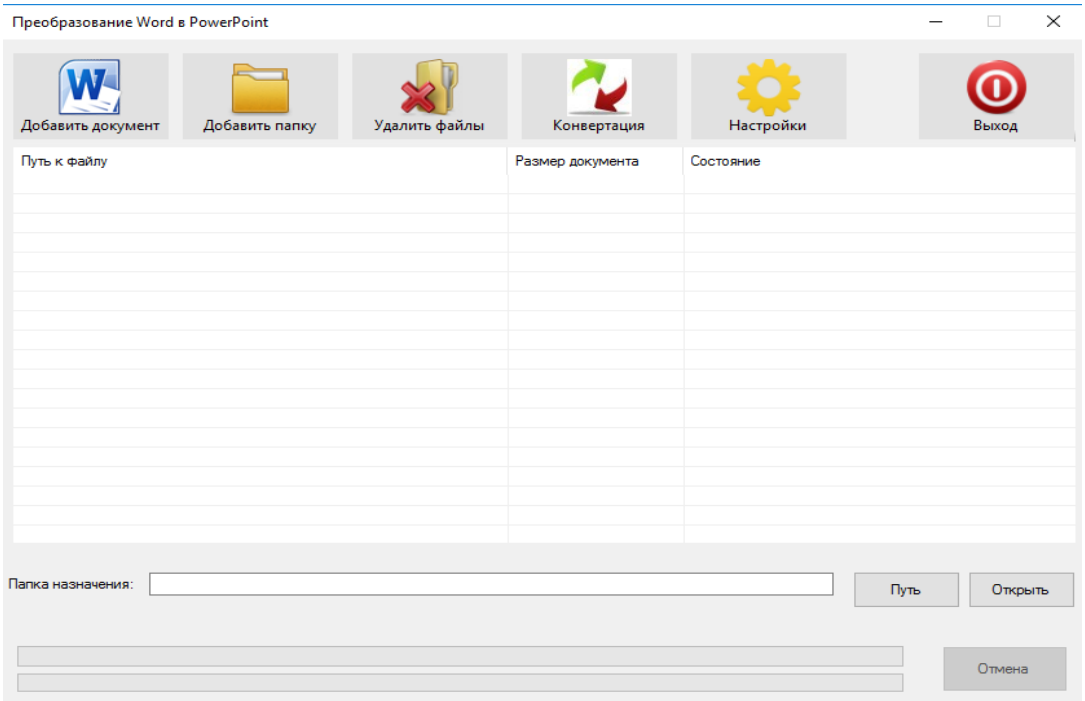

Рисунок 3.11 – Основное окно приложения

Кнопка “Путь” открывает окно, в котором пользователь должен указать папку назначения.

Кнопка “Настройки”, запускает окно, в котором пользователь может изменять шрифт, размер шрифта, цвет и тему оформления (рисунок 3.12).

Для того чтобы хранить настройки по умолчанию для презентаций был реализован класс SettingPresentationDefault. Класс settingPresentation является наследником класса SettingPresentationDefault. В класс settingPresentation добавлены свойства, с помощью которых можно изменять значения полей. Таким образом, когда пользователь меняет настройки, значения присваиваются переменным экземпляра класса SettingPresentation, а затем значения полей используются для задания настроек создаваемой презентации.

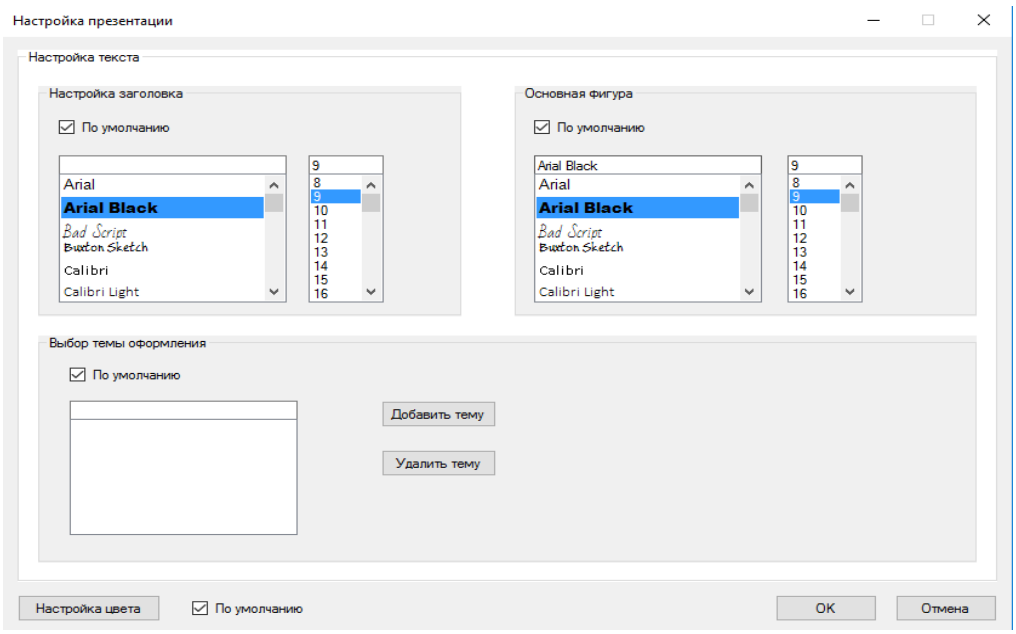

Рисунок 3.12 – Окно настройки приложения

С помощью данного можно поменять следующие настройки: тему оформления, цвет и размер текста.

3.2 Тестирование разработанного приложения

Тестирование – деятельность, выполняемая для оценки и улучшения качества программного обеспечения. Эта деятельность, в общем случае, базируется на обнаружении дефектов и проблем в программных системах. Тестирование разработанного приложения будет проводиться по стратегии черного ящика. При тестировании с помощью стратегии черного ящика сравнивают поведение программы с соответствующими требованиями. Кроме того выявляют ошибки функциональности, которые поддерживает программный продукт, ошибки производимых вычислений. Для проведения тестирование данным методом не требуются знания о внутренней структуре программного обеспечения [21].

Для тестирования применялось модульное тестирование, то есть тестирование каждого отдельного компонента в отдельности. Первый тест выполнялся вручную, была протестирована кнопка “Конвертация”. Для начала был составлен тест-кейс.

Тест-кейс №1. Проверка работоспособности кнопки “Конвертация”.

Шаги.

1. Запустить программу.
2. Выбрать нужный документ.
3. Указать путь назначения.
4. Нажать кнопку “Конвертация”.

Ожидаемый результат: будет создана презентация.

Тестирование показало, что программ работает корректно и в результате работы программы была создана презентация.

Второй тест связан с возможной утечкой памяти (если COM-объекты не будут правильно освобождены). Тестирование, проводилось с помощью NUnit. Фреймворк NUnit позволяет осуществлять модульное тестирование. Была протестирована работа метода MemoryReleasePresentation класса PowerPoint, в случае, когда возникновения исключительной ситуации. MemoryReleasePresentation, отвечает за освобождение COM-объектов приложения PowerPoint. Исключительная ситуация может возникнуть если, например, попытаться применить к презентации не существующую (удаленную) тему оформления.

Тестировался вариант программы без графического интерфейса, для того чтобы было проще осуществить тестирование.

Тест-кейс №2. Проверка работоспособности функции MemoryReleasePresentation.

Шаги.

1. Удалить тему оформления.
2. Запустить программу.
3. Запустить конвертацию.

Ожидаемый результат: процессы приложений Microsoft Word и PowerPoint уничтожены, следовательно, и приложения будут закрыты.

Код для тестирования представлен на рисунке 3.13.

```
[TestFixture]
0 references
class Test
{
    [Test]
    0 references
    public void TestMethod()
    {
        string filePathWord = @"D:\Материалы к слайдам. Часть 1 (1).docx";
        System.IO.FileInfo file = new System.IO.FileInfo(@"D:\Тема4.thmx");
        if(file.Exists)
            file.Delete();
        Word Word1 = new Word();
        Word1.OpenWord(filePathWord);
        bool flag = true;
        for (int i = 0; i < 1000000000; i++) { }
        Process[] word = Process.GetProcessesByName("WINWORD");
        Process[] powerPoint = Process.GetProcessesByName("POWERPNT");
        foreach (Process p in word)
            flag = false;
        foreach (Process p in powerPoint)
            flag = false;
        Assert.IsTrue(flag);
    }
}
```

Рисунок 3.13 – Код для тестирования MemoryReleasePresentation

В данном авто-тесте сначала удаляется тема оформления, затем запускается обработка и далее проверяется, существует ли запущенные процессы приложений Word или PowerPoint. Результат тестирования представлен на рисунке 3.14.

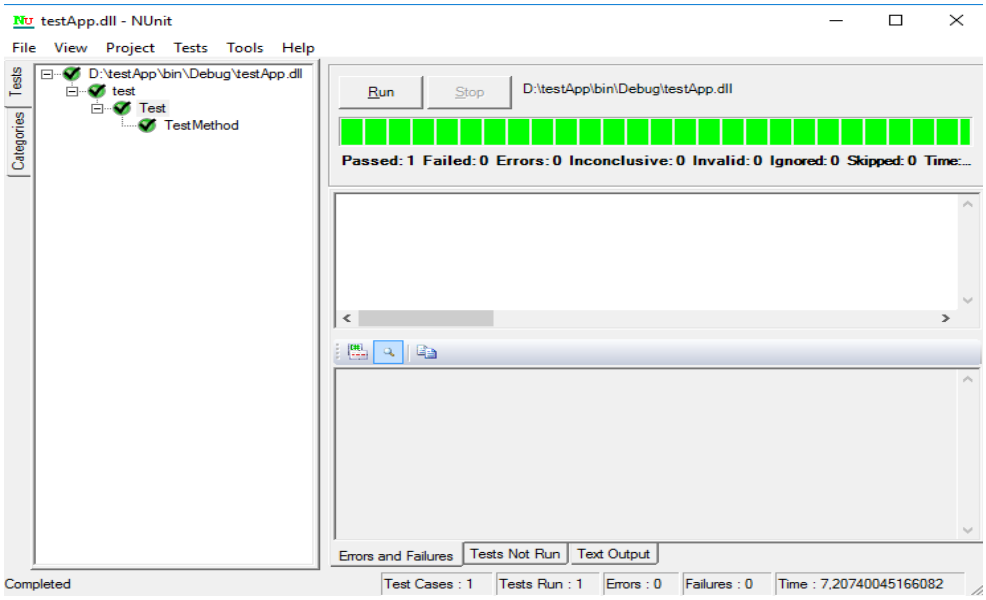

Рисунок 3.14 – Результат тестирования метода MemoryReleasePresentation

Как видно из рисунка 3.14 результат тестирования положительный, а значит, функция MemoryReleasePresentation освободила COM–объекты, и приложения были закрыты.

Вывод по третьей главе

В третьей главе были реализованы основные классы приложения. Так же был реализован графический интерфейс пользователя. Приложения было протестировано на соответствие заявленным функциональным требованиям. Была измерено время работы программы и исходя из результатов можно сделать вывод, что приложение позволяет сократить время, затрачиваемое на создание презентаций.

ЗАКЛЮЧЕНИЕ

В ходе данной выпускной квалификационной работы были выделены объект и предмет исследования, были определенны цели и задачи, проанализированы исходные данные, рассмотрены требования к выходным данным.

Был проведен обзор программных средств, для создания презентации на основе текстового документа, был выбран наиболее подходящий язык программирования и среда разработки. Было реализовано приложение, предназначенное для создания презентаций на основе текстового документа. Приложение было протестировано. Результаты тестирования показали, что приложение соответствует заявленным функциональным требованиям. Обработка 50 слайдов занимает меньше одной минуты. Таким образом, цель и задачи были выполнены и приложение, автоматизирует процесс создания презентаций, тем самым помогая значительно экономить время. В дальнейшем предполагается расширение возможностей программы. Так же необходимо реализовать программу для более новой версии Microsoft Office, чтобы обеспечить стабильную работу с формулами.

Апробация данной работы была представлена в научно-практической конференции «Студенческие Дни науки в ТГУ – 2017», которая проводилась в Тольяттинском государственном университете с 3 по 15 апреля 2017 года на кафедрах институтов в виде семинаров.

Кроме того, апробация работы была представлена и опубликована на III научно-практической всероссийской конференции (школе-семинаре) молодых ученых «Современные исследования в области естественных и технических наук», которая проводилась в Тольяттинском государственном университете с 24 по 25 апреля 2017 года в заочной форме.

Также апробация работы была представлена и опубликована на XIV Всероссийской студенческой конференции «Информационные технологии в современном мире – 2017», которая проводилась организаторами АНО ВО «Гуманитарный университет» 5 мая 2017 года в очно-заочной форме.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Учебники и учебные пособия

1. Албахари Д., Албахари Б. C# 5.0. Справочник. Полное описание языка. 5-е издание / Д. Албахари, Б. Албахари. – М.: Вильямс, 2014. – 1008 с.

2. Аксютин П.А. Опыт построения среды электронного обучения и ее использование для преподавания дисциплины «Информационные технологии» / П.А. Аксютин // Электронное обучение в вузе и в школе: Материалы сетевой международной научно-практической конференции. – СПб.: Астерион - 2014. – С. 28–31.

3. Вугт В.В. OpenXML кратко и доступно/ В.В. Вугт. – Изд-во: Microsoft, 2007. – 109 с.

4. Карпова И. П. Базы данных : курс лекций и материалы для практ. занятий : учеб. пособие для студентов техн. фак. / И. П. Карпова. - Санкт-Петербург : Питер, 2013. - 240 с. : ил. - (Учебное пособие). - Библиогр.: с. 233- 234. - Прил.: с. 211-232. - Алф. указ.: с. 235-240. - ISBN 978-5-496-00546-3 : 418- 60.

5. Макарова Н. В. Информатика: Учебник для вузов. Издательство: Питер, 2013, 576 с.

6. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – М.: Вильямс, 2013. – 1311 с.

Электронные ресурсы

7. Крайнова О. А. Технологии дистанционного обучения [Электронный ресурс] : учебное пособие / О.А. Крайнова. – Электрон. дан. – Тольятти. : Изд-во Тольяттинский государственный университет, 2014. – 124 с. – Режим доступа: http://dspace.tltsu.ru/bitstream/123456789/395/1/%D0%9A%D1%80%D0%B0%D0%B9%D0%BD%D0%BE%D0%B2%D0%B0%201-24-13.pdf

8. Общие сведения о приложениях для Windows [Электронный ресурс]. Режим доступа: https://msdn.microsoft.com/ru-ru/library/5b13a7k4(v=vs.100).aspx

9. Общие сведения об объектной модели Word // MSDN – сеть разработчиков Microsoft [Электронный ресурс]. – Электрон. дан. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/kw65a0we.aspx

10. Организационная структура // Центр новых информационных технологий [Электронный ресурс]. – Электрон. дан. – Режим доступа: http://cnit.tltsu.ru/sites/site.php?s=117&m=29465

11. Технология обучения // Росдистант [Электронный ресурс]. – Электрон. дан. – Режим доступа: http://www.rosdistant.ru/how/

12. The Apache Software Foundation [Electronic resource]. – Electronic data. – Mode of access: http://poi.apache.org/

13. SDK 2.5 Open XML для Office // MSDN – сеть разработчиков Microsoft [Электронный ресурс]. – Электрон. дан. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/office/bb448854.aspx

14. C# // MSDN – сеть разработчиков Microsoft [Электронный ресурс]. – Электрон. дан. – ежим доступа: https://msdn.microsoft.com/ru-ru/library/kx37x362.aspx

15. Java [Электронный ресурс] // Википедия – Электрон. дан. – Режим доступа: https://ru.wikipedia.org/wiki/Java

16. NET APIs to manipulate PowerPoint [Electronic resource]. – Electronic data. – Mode of access: https://www.aspose.com/products/slides/net

17. NET APIs to manipulate Word documents [Electronic resource]. – Electronic data. – Mode of access: https://www.aspose.com/products/words/net

Литература на иностранном языке

18. Dorota Wójcicka-Migasiuk, Arkadiusz Urzędowski, Nizam Omar. Internet tools in education at different levels of teaching [Text] / Dorota Wójcicka-

Migasiuk // Advances in Science and Technology Research Journal. – Volume 8, Issue 24, 2014. – PP. 66-71.

19. Iqbal H. Sarker, Faisal Faruque, Ujjal Hossen and Atikur Rahman. A Survey of Software Development Process Models in Software Engineering [Text]/ Iqbal H. Sarker // International Journal of Software Engineering and Its Applications. - Volume-9, No. 11, 2015. – PP. 55-70.

20. R.S. Madanayake, G.K.A. Dias and N.D. Kodikara. Transforming Simplified Requirement in to a UML Use Case Diagram Using an Open Source Tool [Text] / R.S. Madanayake // International Journal of Computer Science and Software Engineering (IJCSSE). – Volume-6, Issue-3, March 2017. - PP. 61-79.

21. Rasneet Kaur Chauhan, Iqbal Singh. Latest Research and Development on Software Testing Techniques and Tools [Text] / Rasneet Kaur Chauhan // International Journal of Current Engineering and Technology. – Volume 4, No. 4, 2014. – PP. 2368-2372.

22. Vladimir Sulov. Implementation of programing langauages on the platform .net in developing software applications [Text] / Vladimir Sulov // IZVESTIA – Journal of University of Economics – Varna. – No. 1, 2014. – PP. 13-22.

ПРИЛОЖЕНИЕ А. Класс для работы с Word.

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using Microsoft.Office.Interop.Word;
using WordInterop = Microsoft.Office.Interop.Word;
using Office = Microsoft.Office.Core.MsoTriState;
using createPr;
using Word_to_PowerPoint_Converter;

namespace openDoc
{
    2 references
    class DocWord
    {
        delegate void ShowProgressDelegate(bool flag);
        private _Application word;
        private _Document doc;
        private PowerPoint pp;
        private Form1 Forma;
        private string fileNamePresentation;
        private bool error=true;
        2 references
        public bool OpenWord(string filePathWord, Form1 temp, string fileName, SettingPresentation setting)
        {
               try
                {
                System.IO.FileInfo pathWord = new System.IO.FileInfo(filePathWord);
                System.IO.FileInfo copyPathWord = pathWord.CopyTo(System.IO.Path.GetFileNameWithoutExtension(filePathWord), true);
                Forma = temp;
                fileNamePresentation = fileName;
                Object confirmConversions = Type.Missing;
                Object readOnly = false;
                Object addToRecentFiles = true;
                Object passwordDocument = Type.Missing;
                Object passwordTemplate = Type.Missing;
                Object revert = false;
                Object writePasswordDocument = Type.Missing;
                Object writePasswordTemplate = Type.Missing;
                 Object format = Type.Missing;
                 Object encoding = Type.Missing; ;
                 Object oVisible = Type.Missing;
                 Object openConflictDocument = Type.Missing;
                 Object openAndRepair = Type.Missing;
                 Object documentDirection = Type.Missing;
                 Object noEncodingDialog = false;
                 Object xmlTransform = Type.Missing;
                 word = new WordInterop.Application(); //Word
                 word.Visible = false;
                 doc = new Document();
                 doc = word.Documents.Open(copyPathWord.FullName, ref confirmConversions, ref readOnly); // открытие документа
                 documetnProcessing(setting); // запуск обработки документа
                 copyPathWord.Delete();
                 }
                 catch(Exception)
                 {
                   error = false;
                   ShowProgress(error);
                   relaeseMemory();
                 }
            return error;
        }
        1 reference
        public void documetnProcessing(SettingPresentation setting)
        {
                ShowProgress(true);
                Object missing = Type.Missing;
                Range wordRange = doc.Range();
                int i = 1;
                int start, end;
                bool flag = true;
                pp = new PowerPoint(setting,Forma);
                 while (flag) // перебор слайдов
                 {
                     flag = wordRange.Find.Execute("Слайд " + i);
                     start = wordRange.End;
                     wordRange.Start = wordRange.End;
                     flag = wordRange.Find.Execute("Слайд " + (i + 1));
```

```
78                      flag = wordRange.Find.Execute("Слайд " + (i + 1));
79                      if (!flag)
80                          end = doc.Content.End;
81                      else
82                          end = wordRange.Start;
83                      wordRange.SetRange(start, end);
84                      wordRange.Select();
85                      pp.identifObject(wordRange);
86                     if (Forma.getCancel)
87                          break;
88                      i++;
89                  }
90                  ShowProgress(flag);
91                  relaeseMemory();
92          }
            2 references
93          public void relaeseMemory()
94          {
95                try
96               {
97                  string tempFilePath = Forma.filePath;
98                  error = pp.MemoryRelaesePresentation(tempFilePath, fileNamePresentation, error);
99                  doc.Close();
100                 word.Quit();
101                 while (Marshal.ReleaseComObject(word) > 0) { }
102              }
103              catch(Exception)
104              {
105                 doc.Close();
106                 word.Quit();
107                 while (Marshal.ReleaseComObject(word) > 0) { }
108              }
109         }
            4 references
110         public void ShowProgress(bool flag)
111         {
112             if (Forma.InvokeRequired == false)
113                 Forma.changeProgressBar = flag;
114             else
115             {
116                 ShowProgressDelegate showProgres = new ShowProgressDelegate(ShowProgress);
117                 Forma.BeginInvoke(showProgres, flag);
118             }
119         }
120     }
121 }
122
123
```

ПРИЛОЖЕНИЕ Б. Класс для работы с PowerPoint.

```
using System;
using System.IO;
using System.Reflection;
using System.Resources;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using System.Diagnostics;
using Microsoft.Office.Interop.PowerPoint;
using Microsoft.Office.Interop.Word;
using Microsoft.Office.Core;
using Office = Microsoft.Office.Core.MsoTriState;
using WordInterop = Microsoft.Office.Interop.Word;
using PP = Microsoft.Office.Interop.PowerPoint;
using Word_to_PowerPoint_Converter;

namespace createPr
{
    12 references
    class PowerPoint
    {
        private PP.Application Point; // приложениее
        private Presentation presentation; //текущая презентация
        private CustomLayout customLayoutText; //образец слайдов под текст
        private SettingPresentation sp;
        private int numberParagraph; //текущий параграф на слайде
        private int numberSlide = 1; //Текущий слайд
        private Form1 Form;
        private PP.Slide actual;
        1 reference
        public PowerPoint(SettingPresentation setting, Form1 Forma)
        {
            try
            {
                Form = Forma;
                sp = setting;
                Point = new PP.Application(); //  power point
                Point.Visible = Office.msoTrue;
                presentation = Point.Presentations.Add(WithWindow: Office.msoTrue); //Создание презентации
                Point.WindowState = PpWindowState.ppWindowMinimized;
                presentation.ApplyTheme(sp.getPathTheme());// применение темы оформления
                presentation.PageSetup.SlideHeight = 33.867f * 28.35f;
                presentation.PageSetup.SlideWidth = 19.05f * 28.35f;
                presentation.PageSetup.SlideOrientation = MsoOrientation.msoOrientationHorizontal;
                customLayoutText = presentation.SlideMaster.CustomLayouts[PpSlideLayout.ppLayoutText];
                customLayoutText.Shapes[1].TextFrame2.TextRange.Font.Fill.ForeColor.RGB = Color.FromArgb(200, 143, 0).ToArgb();
             }
             catch(Exception)
```

```
50              catch(Exception)
51              {
52                  MemoryRelaesePresentation("","",false);
53              }
54          }
            1 reference
55          public void createSlide()
56          {
57              actual = presentation.Slides.AddSlide(numberSlide, customLayoutText); //создание нового слайда
58              actual.Shapes[1].TextFrame2.VerticalAnchor = MsoVerticalAnchor.msoAnchorMiddle; //выравнивание
59              actual.Shapes[1].TextFrame.TextRange.Font.Size = sp.getSizeTextF1;
60              actual.Shapes[1].TextFrame.TextRange.Font.Name = sp.getFontTextF1;
61              actual.Shapes[2].TextFrame.TextRange.Font.Size = sp.getSizeTextF2;
62              actual.Shapes[2].TextFrame.TextRange.Font.Name = sp.getFontTextF2;
63              actual.Select();
64              numberSlide++;
65          }
            1 reference
66          public void identifObject(WordInterop.Range wordRange) //определяет, что находится в выделенной области
67          {
68              createSlide();
69              numberParagraph = 1;
70              bool flag = false;
71              int countTable = wordRange.Tables.Count;
72              int numberTable = 1;
73              if ( (wordRange.InlineShapes.Count == 1 || wordRange.ShapeRange.Count ==1) && wordRange.OMaths.Count < 1)
74              {
75                  bool controlEquaction = false;
76                  if (wordRange.ShapeRange.Count > 0)
77                      convertToInline(wordRange.ShapeRange[1]);
78                  if (wordRange.InlineShapes[1].Type == WdInlineShapeType.wdInlineShapeEmbeddedOLEObject)
79                  {
80                      if (wordRange.InlineShapes[1].OLEFormat.ClassType != "Equation.3")
81                          controlEquaction = true;
82                  }
83                  if(!controlEquaction)
84                  {
85                      ProcessingImage procImage = new ProcessingImage();
86                      procImage.startProcessing(wordRange, this);
87                  }
88              }
89              else
90              {
91                  foreach (WordInterop.Paragraph paragr in wordRange.Paragraphs)
92                  {
93                      flag = false;
94                      if (paragr.Range.OMaths.Count > 0 || paragr.Range.InlineShapes.Count > 0)
95                      {
96                          flag = true;
```

```
                    ProcessingMath procMath = new ProcessingMath();
                    procMath.starProcessing(paragr, this);
                }
                if (paragr.Range.Tables.Count > 0)
                {
                    flag = true;
                    if (numberTable <= countTable)
                    {
                        ProcessingTable procTable = new ProcessingTable();
                        procTable.startProcessing(paragr.Range.Tables[numberTable], this);
                        numberTable++;
                    }
                }
                if ((paragr.Range.InlineShapes.Count > 0 || paragr.Range.ShapeRange.Count > 0) && wordRange.OMaths.Count < 1)
                {
                    if (paragr.Range.ShapeRange.Count > 0)
                    {
                        int count = paragr.Range.ShapeRange.Count;
                        for (int i = 1; i <= count; i++)
                            convertToInline(paragr.Range.ShapeRange[i]);
                    }
                    for(int i=1; i <= paragr.Range.InlineShapes.Count; i++)
                    {
                        bool inlineShape;
                        bool oleObject;
                        bool smart;
                        bool diagram;
                        inlineShape = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapePicture;
                        oleObject = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapeEmbeddedOLEObject;
                        smart = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapeSmartArt;
                        diagram = paragr.Range.InlineShapes[i].Type == WdInlineShapeType.wdInlineShapeDiagram;
                        if (inlineShape || oleObject || smart || diagram)
                        {
                            flag = true;
                            bool equation = (oleObject ? paragr.Range.InlineShapes[i].OLEFormat.ClassType != "Equation.3" : false);
                            if (equation)
                            {
                                ProcessingImage procImage = new ProcessingImage();
                                procImage.startProcessing(paragr.Range.InlineShapes[i],this);
                                break;
                            }
                        }
                    }
                }
                if (!flag)
                {
                    ProcessingText procText = new ProcessingText();
                    procText.starProcessing(paragr, this);
                }
            }
        }
```

```
            if (actual.Shapes.Count > 2 && actual.Shapes[2].Type == MsoShapeType.msoPlaceholder)
            {
                if (actual.Shapes[2].TextFrame.TextRange.Text.Trim().Length < 1)
                {
                    actual.Shapes[2].TextFrame2.DeleteText();
                    actual.Shapes[2].Delete();
                    finalProcess();
                }
            }
        }
        2 references
        private void convertToInline(WordInterop.Shape shapeRange)
        {
            bool inlineShape;
            bool oleObject;
            bool smart;
            bool diagram;
            inlineShape = shapeRange.Type == MsoShapeType.msoPicture;
            oleObject = shapeRange.Type == MsoShapeType.msoEmbeddedOLEObject;
            smart = shapeRange.Type == MsoShapeType.msoSmartArt;
            diagram = shapeRange.Type == MsoShapeType.msoDiagram;
            if (inlineShape || oleObject || smart || diagram)
                shapeRange.ConvertToInlineShape();
        }
        1 reference
        private void finalProcess()
        {
            PP.Shape shps = presentation.Slides[presentation.Slides.Count].Shapes[presentation.Slides[presentation.Slides.Count].Shapes.Count];
            if (shps.Type == MsoShapeType.msoTable)
            {
                int row = shps.Table.Rows.Count;
                shps.Width = 25.52f * 28.35f;
                shps.Top = 5f * 28.35f;
                shps.Left = 5f * 28.35f;
                for (int i = 1; i <= row; i++)
                {
                    int columns = shps.Table.Rows[i].Cells.Count;
                    for (int j = 1; j <= columns; j++)
                    {
                        PP.Shape tempTable = shps.Table.Rows[i].Cells[j].Shape;
                        tempTable.TextFrame2.TextRange.Font.Size = 22;
                        tempTable.TextFrame2.TextRange.ParagraphFormat.SpaceWithin = 1.0f;
                        tempTable.TextFrame2.VerticalAnchor = MsoVerticalAnchor.msoAnchorMiddle;
                        tempTable.Fill.ForeColor.RGB = Color.FromArgb(255, 255, 255).ToArgb();
                        tempTable.Fill.Visible = Office.msoFalse;
                        if (tempTable.TextFrame2.TextRange.Font.Bold == MsoTriState.msoTrue)
                        {
                            tempTable.TextFrame2.TextRange.Font.Bold = MsoTriState.msoFalse;
                            tempTable.TextFrame2.TextRange.Font.Fill.ForeColor.RGB = Color.FromArgb(63, 63, 63).ToArgb();
                        }
                        else
                            tempTable.TextFrame2.TextRange.Font.Fill.ForeColor.RGB = Color.FromArgb(112, 112, 112).ToArgb();

                        while ((j + 1) <= columns && shps.Table.Cell(i, j + 1).Shape.TextFrame.TextRange.Text ==
                            shps.Table.Cell(i, j).Shape.TextFrame.TextRange.Text)
                            j++;
                    }
                }
                float slideHeight = presentation1.PageSetup.SlideHeight;
                float shapeHeigth = actual.Shapes[1].Height;
                if ((slideHeight - 2.5f * 28.35f) - (shapeHeigth + 0.5f * 28.35f) > shps.Height || (slideHeight - 2.5f * 28.35f) - (shapeHeigth + 0.5f * 28.35f) < shps.Height)
                    shps.Height = (slideHeight - 2.5f * 28.35f) - (shapeHeigth + 0.5f * 28.35f);
            }
        }
        1 reference
        public bool SavePresentation(string filePath, string fileNamePresentation)
        {
            bool errortemp;
            try
            {
                    string temp = System.IO.Path.Combine(filePath, fileNamePresentation + ".pptx");
                    while (System.IO.File.Exists(temp))
                    {
                        Random rnd = new Random();
                        int value = rnd.Next();
                        temp = System.IO.Path.Combine(filePath, fileNamePresentation + value.ToString() + ".pptx");
                    }
                    presentation.SaveAs(temp);
                    errortemp = true;
            }
            catch(COMException)
            {
                errortemp=false;
                MessageBox.Show(
                   "Невозможно сохранить презентации в выбранной директории",
                   "Сообщение",
                   MessageBoxButtons.OK,
                   MessageBoxIcon.Information);
            }
            return errortemp;
        }
        2 references
        public bool MemoryRelaesePresentation(string filePath, string fileNamePresentation, bool error)
        {
                if(error)
                  error = SavePresentation(filePath, fileNamePresentation);
                if (actual != null)
                    Marshal.FinalReleaseComObject(actual);
                if (customLayoutText != null)
                    Marshal.FinalReleaseComObject(customLayoutText);
                presentation.Close();
                Marshal.FinalReleaseComObject(presentation);
                Point.Quit();
                Marshal.FinalReleaseComObject(Point);
```

```
246                 Marshal.FinalReleaseComObject(Point);
247                 GC.Collect();
248                 GC.WaitForPendingFinalizers();
249                 return error;
250             }
                5 references
251             public PP._Application application
252             {
253                 get
254                 {
255                     return Point;
256                 }
257             }
                5 references
258             public _Presentation presentation1
259             {
260                 get
261                 {
262                     return presentation;
263                 }
264             }
                48 references
265             public int numberParagraph1
266             {
267                 get
268                 {
269                     return numberParagraph;
270                 }
271                 set
272                 {
273                     numberParagraph = value;
274                 }
275             }
                163 references
276             public PP.Slide actual1
277             {
278                 get
279                 {
280                     return actual;
281                 }
282             }
                40 references
283             public SettingPresentation sp1
284             {
285                 get
286                 {
287                     return sp;
288                 }
289             }
```

```
276         public PP.Slide actual1
277         {
278             get
279             {
280                 return actual;
281             }
282         }
            40 references
283         public SettingPresentation sp1
284         {
285             get
286             {
287                 return sp;
288             }
289         }
            2 references
290         public Form1 forma
291         {
292             get
293             {
294                 return Form;
295             }
296         }
297     }
298 }
299
300
301
```