**Задание «Автоматизация Word».** В среде объектно-ориентированного программирования *Microsoft Visual Studio* разработать программу в режиме *Windows Forms Application* на языке *Visual C#*, представляющую собой средство взаимодействия с текстовым файлом, необходимым для настройки формата надписей, заносимых в документ *Microsoft Office Word.*

**Рабочее задание:**

1. Создать и настроить графический пользовательский интерфейс, взаимодействующий с текстовым файлом (*\*.txt*), а также с модулем текстового редактора *Microsoft Office Word*.

2. Создать текстовый файл. Текстовый файл исходно пуст, но нацелен хранить *M* строк с разделителями вида:

*param1;param2;param3;…;paramN*

Каждая строка – набор значимых параметров, записанных в порядке, предусмотренных автором-разработчиком графического пользовательского интерфейса. Количество параметров *N* так же определяется автором-разработчиком.

Количество строк *M* определяется количеством абзацев на титульном листе отчёта к работе в формате *Microsoft Office Word*. Например:

а. Наименование ведомства (Министерство транспорта Российской Федерации);

б. Регалии вуза (Федеральное государственное бюджетное образовательное учреждение высшего образования);

в. Наименование вуза («Российский университет транспорта» (РУТ (МИИТ));

г. Наименование института (Институт транспортной техники и систем управления);

д. Наименование кафедры (Кафедра «Управление и защита информации»);

е. Вид документа (Отчёт / Реферат / Курсовая работа / Курсовой проект);

ж. Вариация связки (по практике / по);

и. Вид учебного занятия (Учебной практике / Производственной практике / Лабораторной работе / Практическому занятию / Индивидуальному заданию);

к. Вариация связки, если необходима (по дисциплине);

л. Наименование дисциплины, если необходимо («Алгоритмизация и технологии программирования»);

м. Вариация связки, если необходима (на тему / представлено решение задачи);

н. Строка «Выполнил: ст. гр.» с указанием номера учебной группы;

о. Ф.И.О. автора-составителя документа;

п. Номер варианта;

р. Строка «Проверил: » с указанием регалий и Ф.И.О. проверяющего лица.

с. Строка формата «Город – год».

Например, *param1* – строка текста, выводимая в абзац документа *Microsoft Office Word; param2* – семейство/наименование шрифта (*Times New Roman*); *param3* – размер шрифта (15 пт) и так далее.

3. При загрузке графического пользовательского интерфейса все параметры из файла считываются в строковый массив, если файл не пуст, и изменяют состояние элементов управления, размещённых на графическом пользовательском интерфейсе. Если файл пуст, то все элементы управления графического пользовательского интерфейса остаются в исходном, нулевом состоянии.

4. Должна быть предусмотрена кнопка «Сохранить», записывающая в файл выставленные или изменённые на графическом пользовательском интерфейсе значения параметров.

5. Должна быть предусмотрена кнопка «Создать», инициирующая генерацию титульного листа в формате *Microsoft Office Word.*

**Цель работы (одна из возможных формулировок)**: освоить навыки подключения внешних модулей и их использования в разрабатываемых программах; освоить навыки программируемой настройки параметров документов *Microsoft Office Word*.

**В качестве результата выполненной работы сдаётся**:

1. Архив *\*.rar* или *\*.zip*, содержащий разработанное решение (*solution*). Для отправки архива по электронной почте рекомендуется использовать сервисы файлового обмена, генерирующие ссылки на файлы (например, *Google Drive*, *Яндекс.Диск*, *Облако Mail.Ru* и тому подобные), поскольку протоколы электронной почты часто автоматически отклоняют архивы, содержащие исполняемые (*exe*) файлы.

2. Выгруженный программой титульный лист в формате *Microsoft Office Word.*

3. Отчёт в формате *Microsoft Office Word* по выполненной работе.

**Требования к блок-схемам алгоритмов:**

1. Изображать блок-схему алгоритма требуется только по содержательной части программного обеспечения, отвечающей за настройку документа *Microsoft Office Word*. Настройку графического пользовательского интерфейса программы описывать блок-схемами алгоритмов не требуется.

2. В том случае, если содержательная часть разбита на методы, необходимо составить сначала обобщённую блок-схему, иллюстрирующую последовательность вызова методов, а затем каждый из методов проиллюстрировать отдельной, детализированной блок-схемой.

**Памятка программисту №1. Обнаружение и перенос библиотеки *Word:***

1. В главном меню раскрыть содержимое пункта «Проект» и перейти к диалоговому окну «Добавить ссылку…» (Рисунок 1).

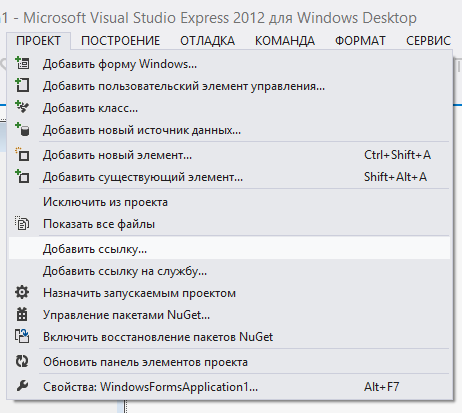


Рисунок 1 – Содержимое пункта меню «Проект» (*Visual Studio 2012*)

2. В открывшемся диалоговом окне переключиться на вкладку «*COM*» (Рисунок 2).

3. Найти в списке компонент, содержащий сочетание слов «*Microsoft Word*» (Рисунок 2).

4. Определить путь к соответствующей библиотеке по всплывающему *tool-tip*’у или в одном из столбцов диалогового окна (Рисунок 2).

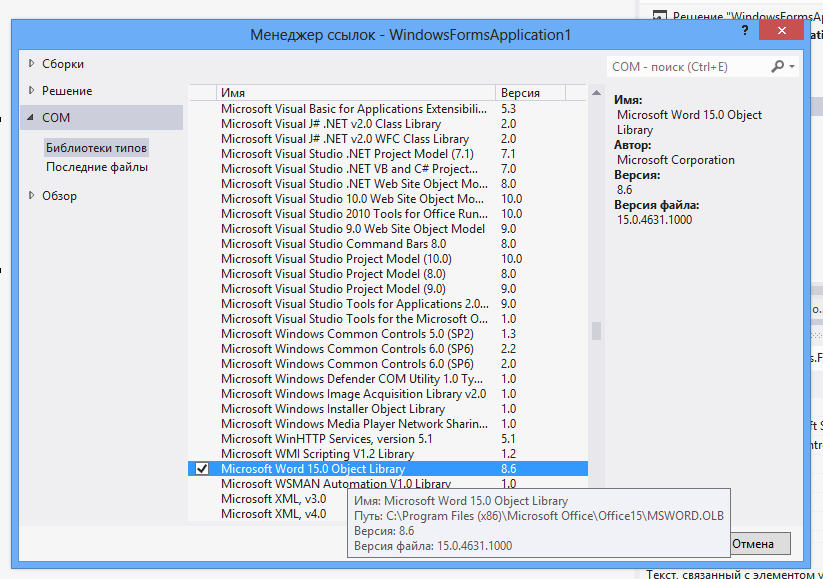


Рисунок 2 – Диалоговое окно – менеджер ссылок (*Visual Studio 2012*)

Нужным файлом может оказаться «*MSWORD.OLB*», который находиться, например, в *«\Program Files\Microsoft Office\Office14\*» или *«\Program Files\Microsoft Office\Office15\*».

Важно, чтобы численное значение, указанное после слова «*Office*» (это кодовая версия *Microsoft Office*), было не старше значения, которое будет указано на компьютере, где планируется последующая эксплуатация приложения.

В том случае, если известно, что на компьютере конечного пользователя установлена более ранняя версия *Microsoft Office* по сравнению с той, которая установлена на компьютере разработчика, то не лишним шагом будет предварительное копирование в директорию проекта нужного файла именно с компьютера конечного пользователя, а не с компьютера разработчика.

Полезная информация относительно соответствия рабочих версий пакетов *Microsoft Office* следующая:

***Office 11 – Microsoft Office 2003;***

***Office 12 – Microsoft Office 2007;***

***Office 14 – Microsoft Office 2010;***

***Office 15 – Microsoft Office 2013.***

5. Найти по определённому в диалоговом окне пути библиотеку «*MSWORD.OLB*» (Рисунок 3).

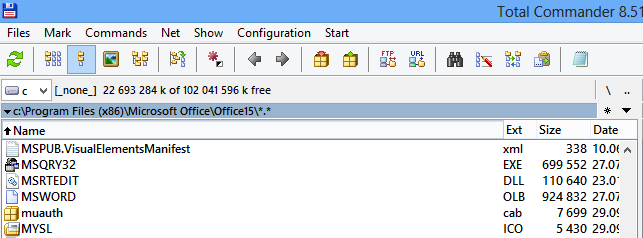


Рисунок 3 – Поиск библиотеки *\*.olb* в файловом менеджере (*Total Commander*) в директории *Microsoft Office*

6. Скопировать библиотеку в директорию с разрабатываемым проектом/решением (*solution*), как показано на Рисунке 4.

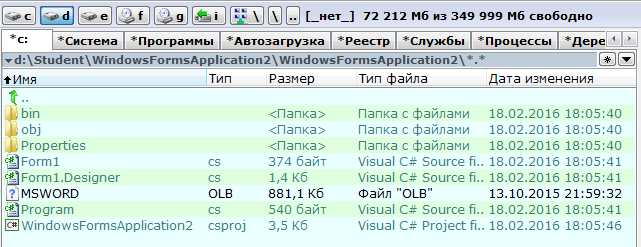


Рисунок 4 – Копирование библиотеки *\*.olb* в файловом менеджере (*Total Commander*) в директорию с проектом *Windows Forms Application*

**Памятка программисту №2. Подключение библиотеки *Microsoft Word* к решению (*solution*):**

1. В главном меню раскрыть содержимое пункта «Проект» и перейти к диалоговому окну «Добавить ссылку…» (Рисунок 1).

2. В открывшемся диалоговом окне переключиться на вкладку «Обзор» (Рисунок 5).

3. Выбрать библиотеку *Microsoft Office Word* и подтвердить выбор нажатием на кнопку «*ОК»* (Рисунок 5).

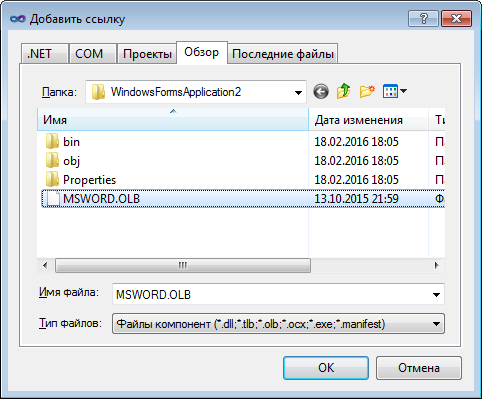


Рисунок 5 – Диалоговое окно – добавить ссылку (*Visual Studio 2010*)

4. В обозревателе решений (Рисунок 6) раскрыть директорию «Ссылки» и убедиться в подключении указанной ранее библиотеки (*Microsoft.Office.Core,* *Microsoft.Office.Interop.Word, VBIDE*).

|  |  |
| --- | --- |
| **до** | **после** |

Рисунок 6 – Обозреватель решений, директория «Ссылки»

5. Прописать в разделе подключаемых к решению модулей ***using*** следующий: ***Microsoft.Office.Interop.Word.***

Для удобства обращения к модулю присвоить его переменной, например, *Word* или *W* (Рисунок 7).

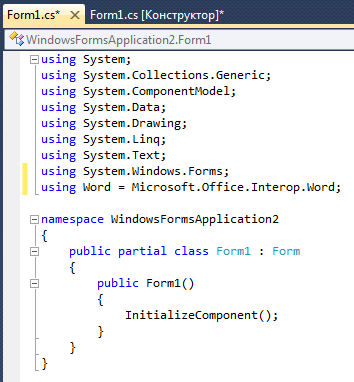


Рисунок 7 – Фрагмент код главной формы проекта *Windows Forms Application*

**Памятка программисту №3. Работа с *Microsoft Office Word* программным способом**

1. Обобщённая конструкция для работы с *Microsoft Office Word* может быть записана следующим образом (Рисунок 8):

а) В разделе объявления используемых модулей ***using*** прописать

***System.Reflection****;*

он необходим для обращения к *Missing* (отсутствующему параметру).

б) Создать объект под отсутствующие параметры *Microsoft Office* *Word*

***object ObjMissing = Missing.Value.***

в) Поместить блок кода для отлова ошибок ***try … catch … finally … .***

г) Создать переменную под приложение *Microsoft Office* *Word* (серая подложка):

***Word.\_Application ObjWord = new Word.Application().***

д) Создать метод, в который в качестве параметров необходимо передавать *Word*-приложение и отсутствующее значение.

е) Закрыть *Word*-приложение методом *Quit().*

ж) В разделе *catch* вызвать побуждающее сообщение на случай возникновения непредвиденной ошибки в процессе работы с *Microsoft Office Word.*

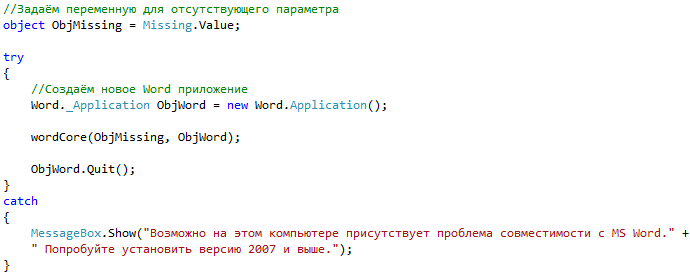
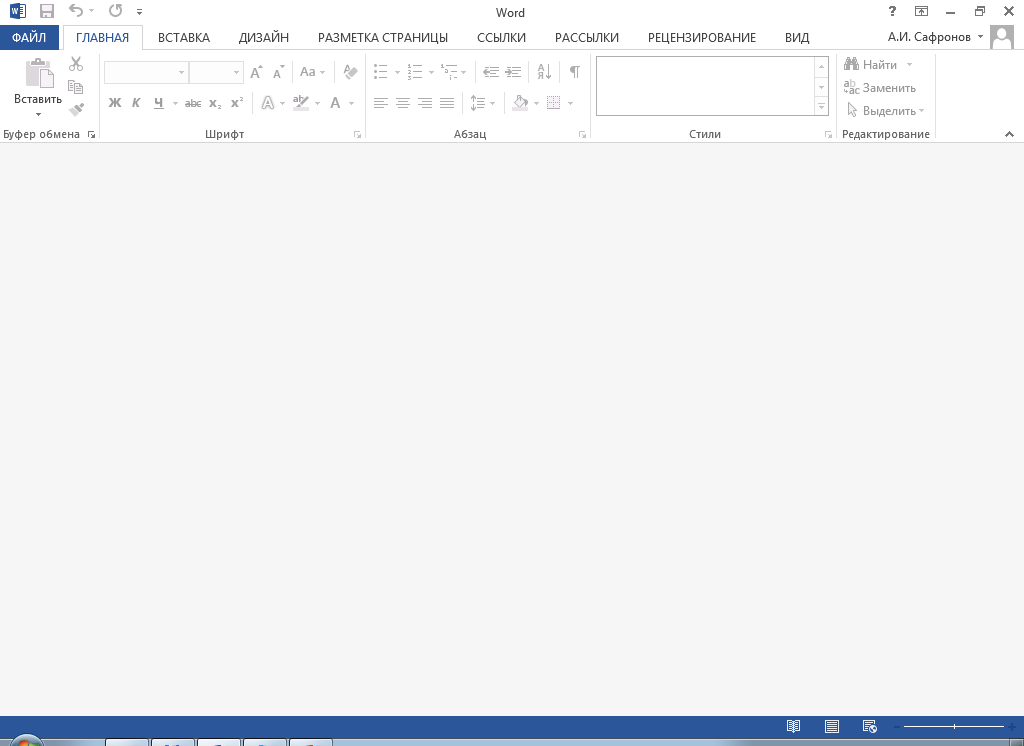


Рисунок 8 – Код обобщённой конструкции для взаимодействия с *Microsoft Office Word*

Пониманию процесса создания приложения *Microsoft Office* *Word* могут способствовать следующие иллюстрации (Рисунок 9-10):



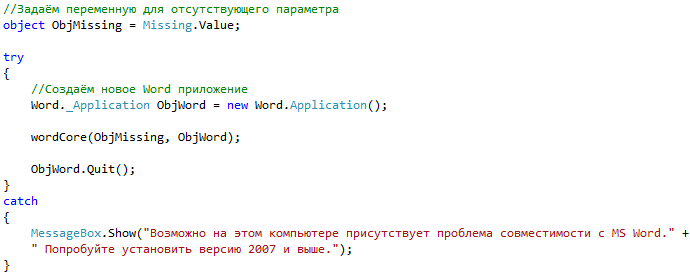


Рисунок 9 – Серая подложка приложения *Microsoft Office Word* и код, соответствующий генерации/вызову подложки

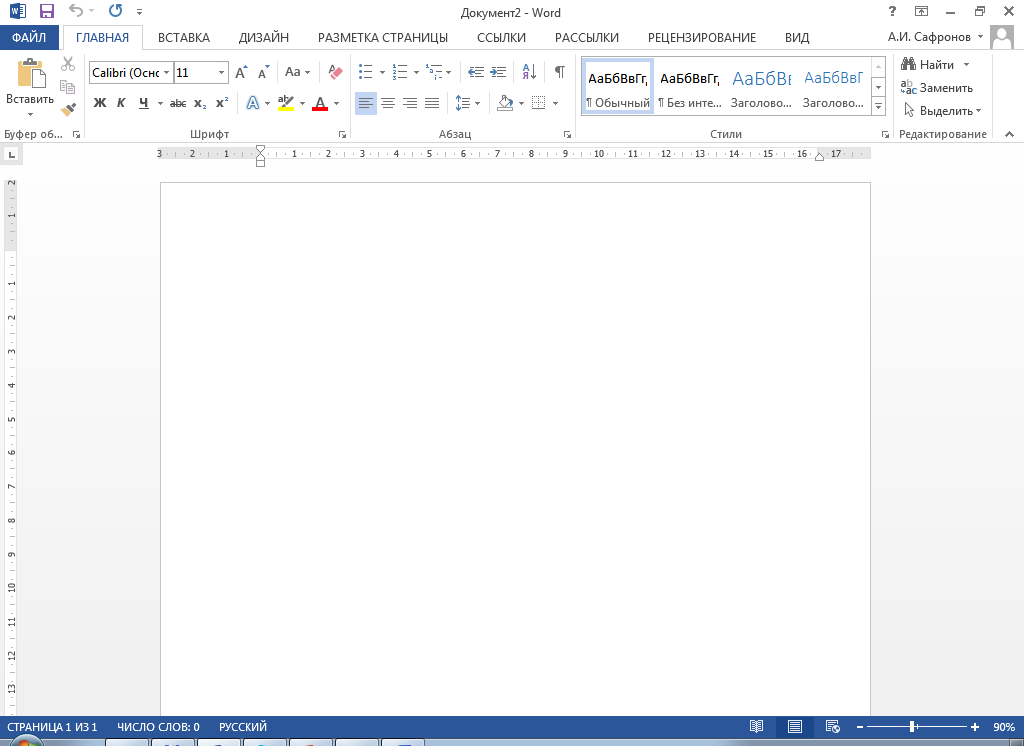




Рисунок 10 – Документ *Microsoft Office Word* и код, соответствующий добавлению и активации документа

Создание нового параграфа/абзаца текста как объекта реализуется посредством нижеследующей кодовой конструкции (Рисунок 11):



Рисунок 11 – Создание нового абзаца в документа *Microsoft Office Word*

В ней сначала указывается объект-диапазон (*ObjRange*), который находится по скрытой закладке (*Bookmark*), заданной по умолчанию в самом конце документа *Microsoft Office Word*. Эта закладка называется «*\\endofdoc*» (конец документа) и для удобства использования её рационально записать в строковую переменную с аналогичным именем *EndOfDoc*.

Новый параграф/абзац добавляется методом *Add()* в документ *Microsoft Office Word* (*ObjDoc*), как его содержимое (*Content*), и, вместе с тем, как один из абзацев (*Paragraphs*) в выделенном диапазоне (*ObjRange*).

Пониманию структуры документа *Microsoft Office* *Word* может способствовать следующая схема (Рисунок 12):



Рисунок 12 – Объектно-ориентированная структура *Microsoft Office Word*

Процедура, в рамках которой документ *Microsoft Office Word* наполняется содержимым, может, в общем виде, быть представлена следующей конструкцией:

а) Создаётся объект, хранящий закладку, которой отмечено окончание документа *Microsoft Office Word*

***object EndOfDoc = “\\endofdoc”.***

б) Для экономии ресурсов *Word*-приложение делается невидимым

**ObjWord.Visible = false.**

в) В *Word*-приложении создаётся новый документ

**Word.\_Document ObjDoc =**

**ObjWord.Application.Documents.Add().**

г) Созданный документ активируется (включается в работу)

**ObjDoc.Activate().**

д) Создаётся объект, отвечающий за выбранную область документа. В него необходимо поместить рабочий диапазон документа *Microsoft Office Word*, находящийся в конце

***object ObjRange* = ObjDoc.Bookmarks.get\_Item(ref EndOfDoc).Range;**

При необходимости перейти к новому абзацу с текстом конструкция прописывается повторно, но уже без объявления типа *object*.

е) В указанной рабочей области создаётся новый абзац

**Word.Paragraph ObjParagraph = ObjDoc.Content.Paragraphs.Add(ref ObjRange);**

ж) Для размещения в абзаце какого-либо текста прописывается

***ObjParagraph.Range.Text = “Какой-либо текст”;***

з) Выполняется сохранение файла (например, в той же директории, где размещён исполняемый (*exe*) файл)

***ObjDoc.SaveAs(Application.StartupPath + @”\” + Name + “.docx”);***

и) Закрыть документ *Word*

***ObjDoc.Close();***

В переменную *Name* в любой удобный момент времени, предшествующий сохранению, необходимо записать название файла, который будет храниться на жёстком диске или любом другом носителе информации.

**Памятка программисту №3. Пример работы с абзацами**

Пример кода метода для вставки нужного количества отступов перед абзацем представлен на Рисунке 13:

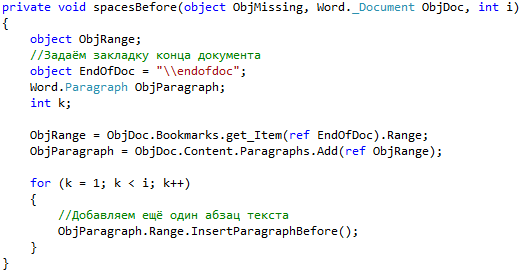


Рисунок 13 – Пример метода вставки i штук пустых абзацев перед последним созданным в *Microsoft Office Word*

Пример кода метода для вставки нужного количества отступов после абзаца представлен на Рисунке 14:

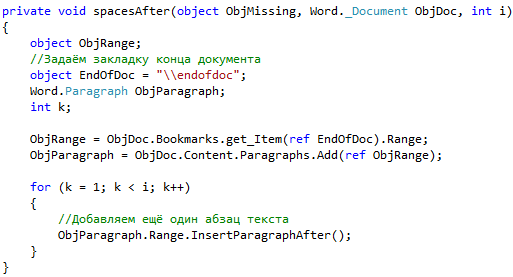


Рисунок 14 – Пример метода вставки i штук пустых абзацев после последнего созданного в *Microsoft Office Word*

Далее следуют некоторые примеры оформления абзацев, которые могут быть полезны для выполнения данного задания. На Рисунке 15 представлен фрагмент титульного листа индивидуального плана преподавателя. Равнозначный по оформлению текст, следующий друг за другом по строкам, может быть вставлен с использованием метода *InsertAfter()* в сочетании с *InsertParagraphAfter()*, как показано на Рисунке 16.

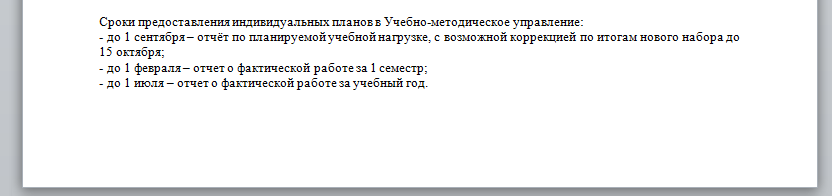


Рисунок 15 – Пример работы с равноценным по оформлению текстом

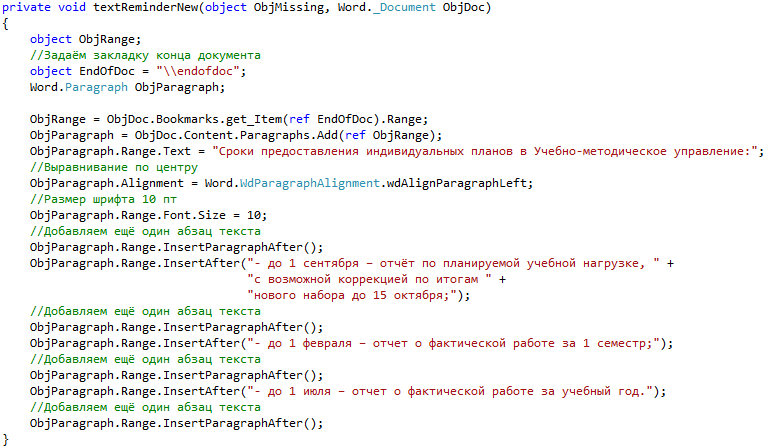


Рисунок 16 – Фрагмент кода программы для примера работы с равноценным по оформлению текстом

Для удобства чтения составленного кода и последующего формирования отчётной документации по работе рекомендуется заключать настройку значащих фрагментов документа в отдельные методы, как показано на Рисунке 17.

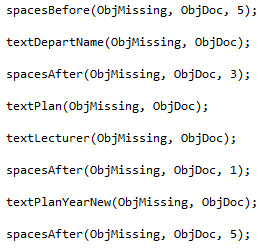


Рисунок 17 – Фрагмент кода программы, в которой реализовано обобщение алгоритма посредством разделения на отдельные методы

Представленному на Рисунке 17 коду соответствует иллюстрация фрагмента документа *Microsoft Office Word* (Рисунок 18).

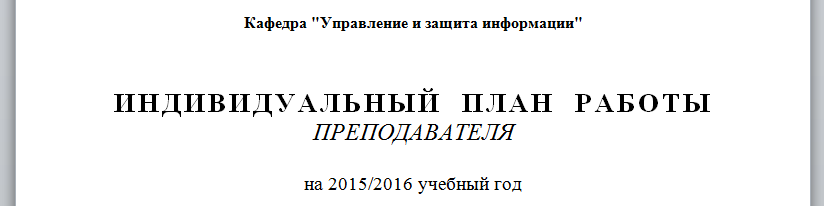


Рисунок 18 – Фрагмент документа *Microsoft Office Word*, на котором представлено принципиально различное оформление текста, расположенного в нескольких соседних абзацах

На последующих Рисунках 19-22 представлены детализированные фрагменты кода методов, упомянутых на Рисунке 17.

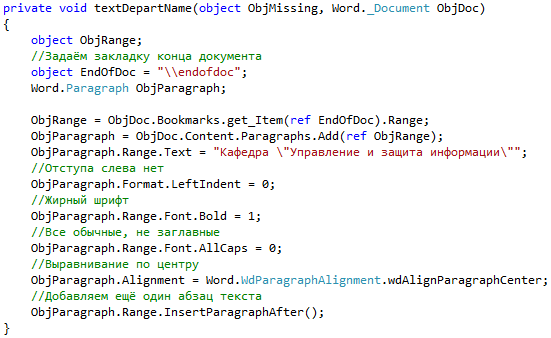


Рисунок 19 – Фрагмент кода, соответствующий форматированию наименования кафедры

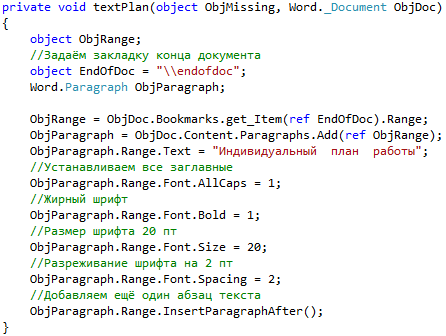


Рисунок 20 – Фрагмент кода, соответствующий форматированию надписи «Индивидуальный план работы»

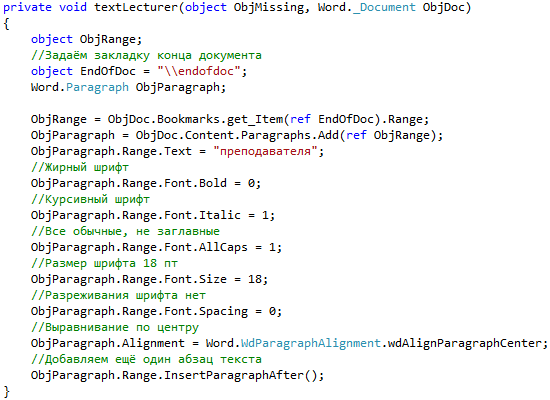


Рисунок 21 – Фрагмент кода, соответствующий форматированию надписи «Преподавателя»

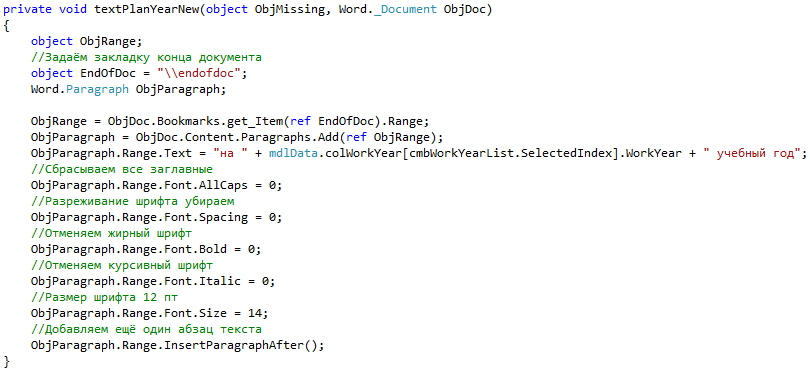


Рисунок 22 – Фрагмент кода, соответствующий форматированию надписи с указанием учебного года

**Памятка программисту №4. Пример работы с таблицами**

Большинство современных графических пользовательских интерфейсов размечаются с использованием гибких табличных структур без обрамления. На Рисунке 23 представлен один из простых примеров подобного рода разметки в документе *Microsoft Office Word.*

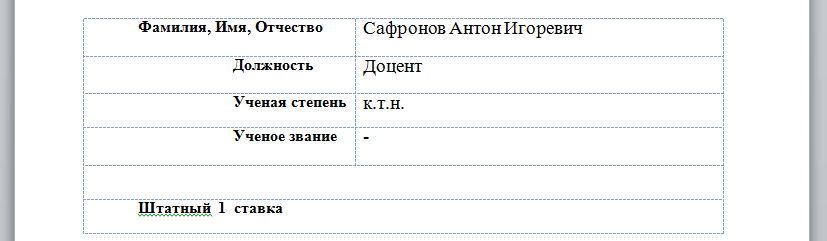


Рисунок 23 – Пример табличной разметки на титульном листе

Далее на Рисунках 24-26 представлен метод, в котором индивидуально настраивается каждая из ячеек таблицы, представленной на Рисунке 23.

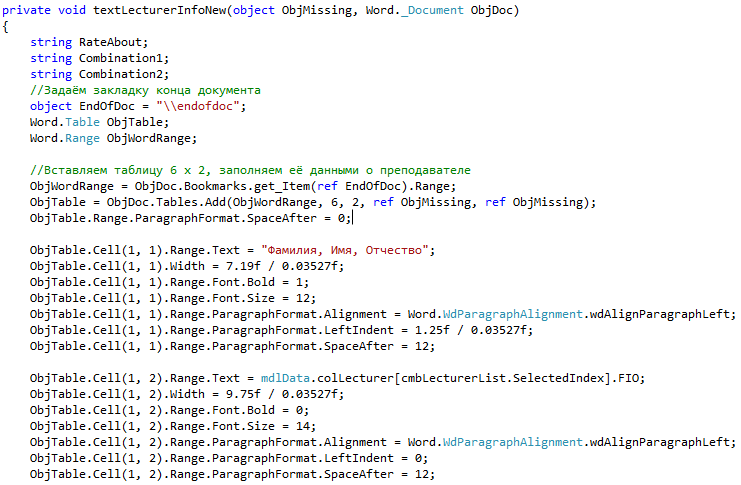


Рисунок 24 – Примерный код табличной разметки в документе *Microsoft Office Word* (начало)

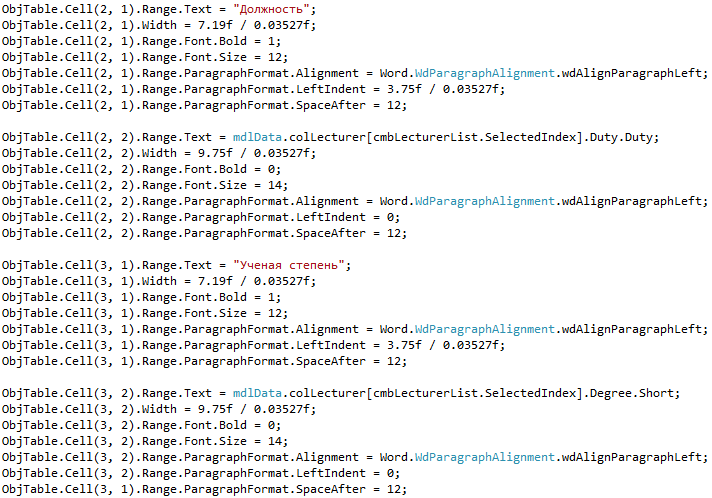


Рисунок 25 – Примерный код табличной разметки в документе *Microsoft Office Word* (продолжение)

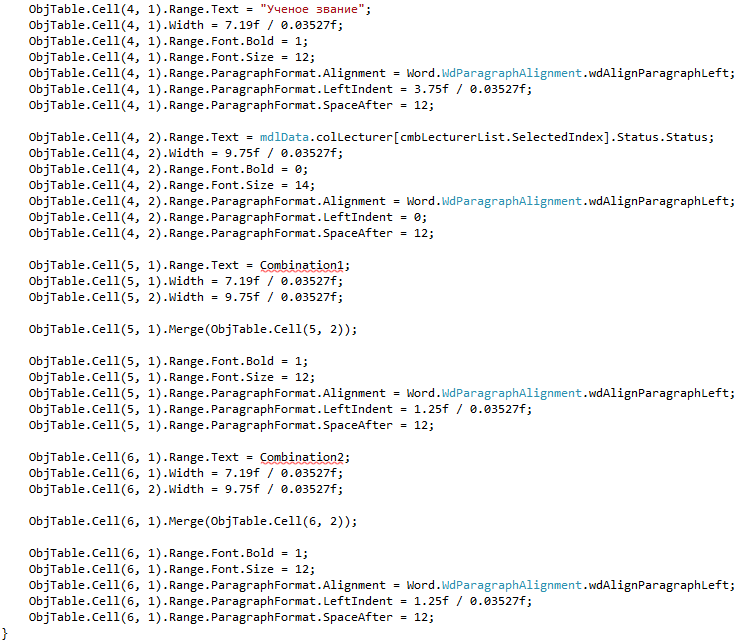


Рисунок 26 – Примерный код табличной разметки в документе *Microsoft Office Word* (окончание)

**Памятка программисту №5. Пример создания разрыва страницы**

В отдельных случаях, когда документ *Microsoft Office Word* должен содержать несколько страниц с разделами, строго позиционированными на заранее определённых листах без каких-либо сдвижек при наполнении этих разделов текстом, рекомендуется использовать перенос курсора на следующую страницу посредством ввода разрыва (Рисунок 27).

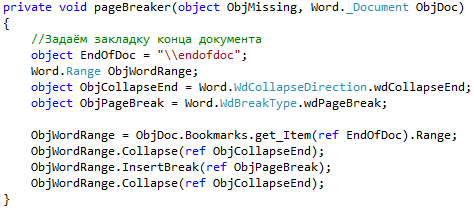


Рисунок 27 – Примерный код метода, реализующего разрыв страницы

**Памятка программисту №6. Иные полезные настройки при работе с *Microsoft Office Word***

После того, как документ *Microsoft Office Word* создан и активирован, можно обратиться к его свойству *PageSetup*. Это очень полезное свойство, если понимать, чему соответствует каждый из его параметров. Соответствие проиллюстрировано на Рисунках 28-31.

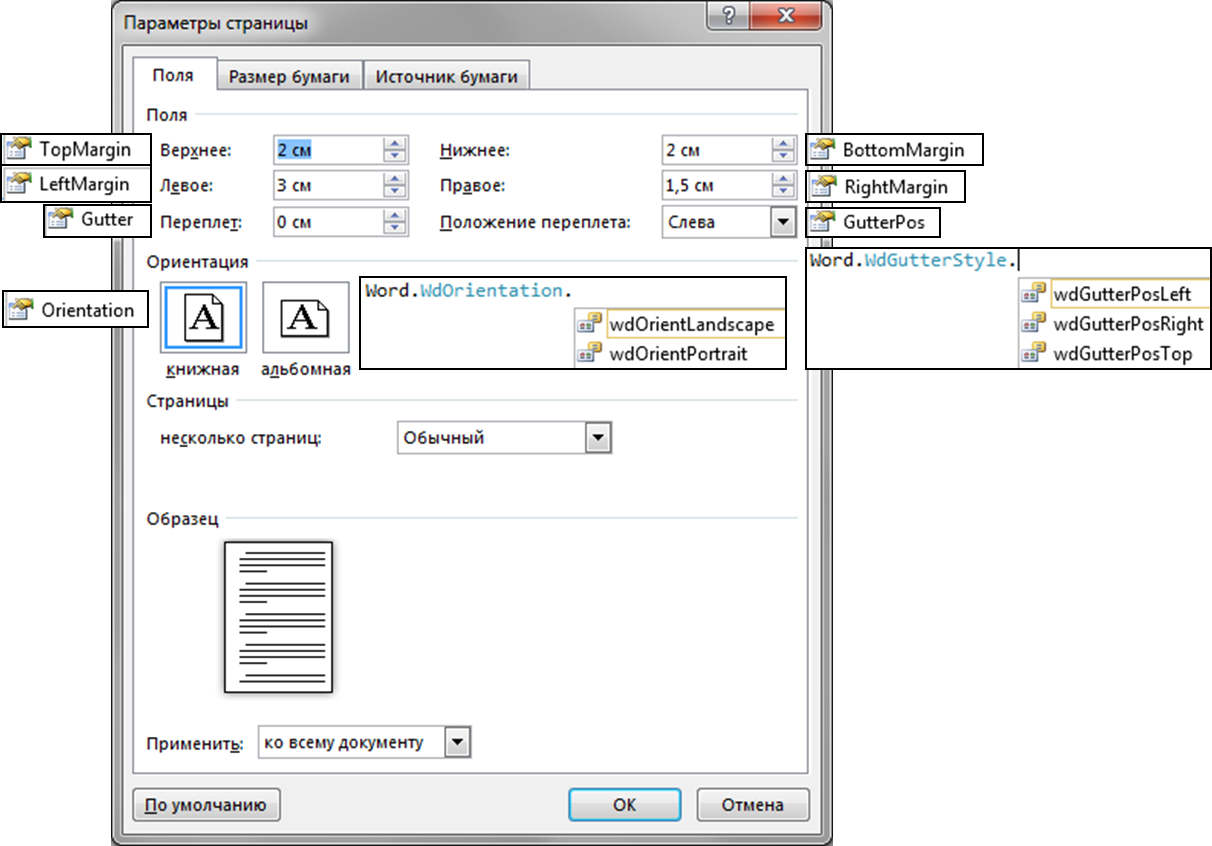


Рисунок 28 – Настройка полей документа *Microsoft Office Word* программным способом

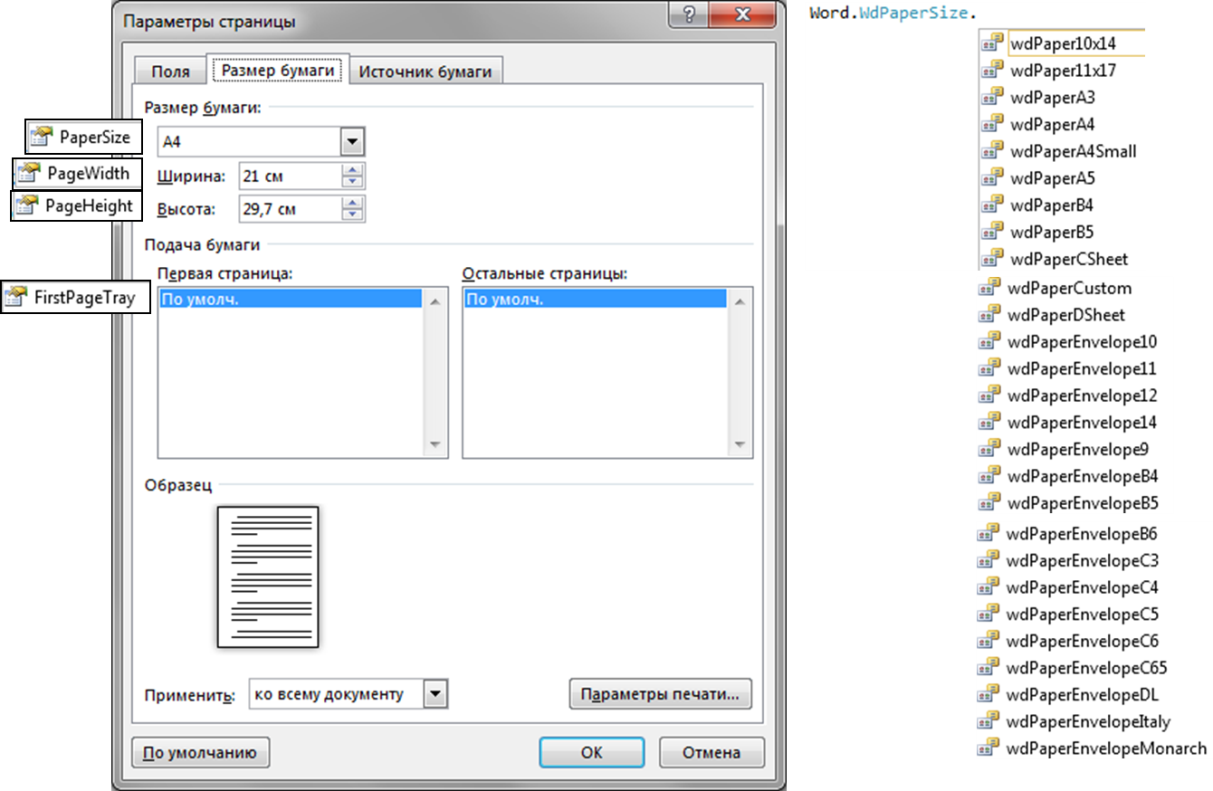


Рисунок 29 – Настройка параметров бумажных носителей для документа *Microsoft Office Word* программным способом

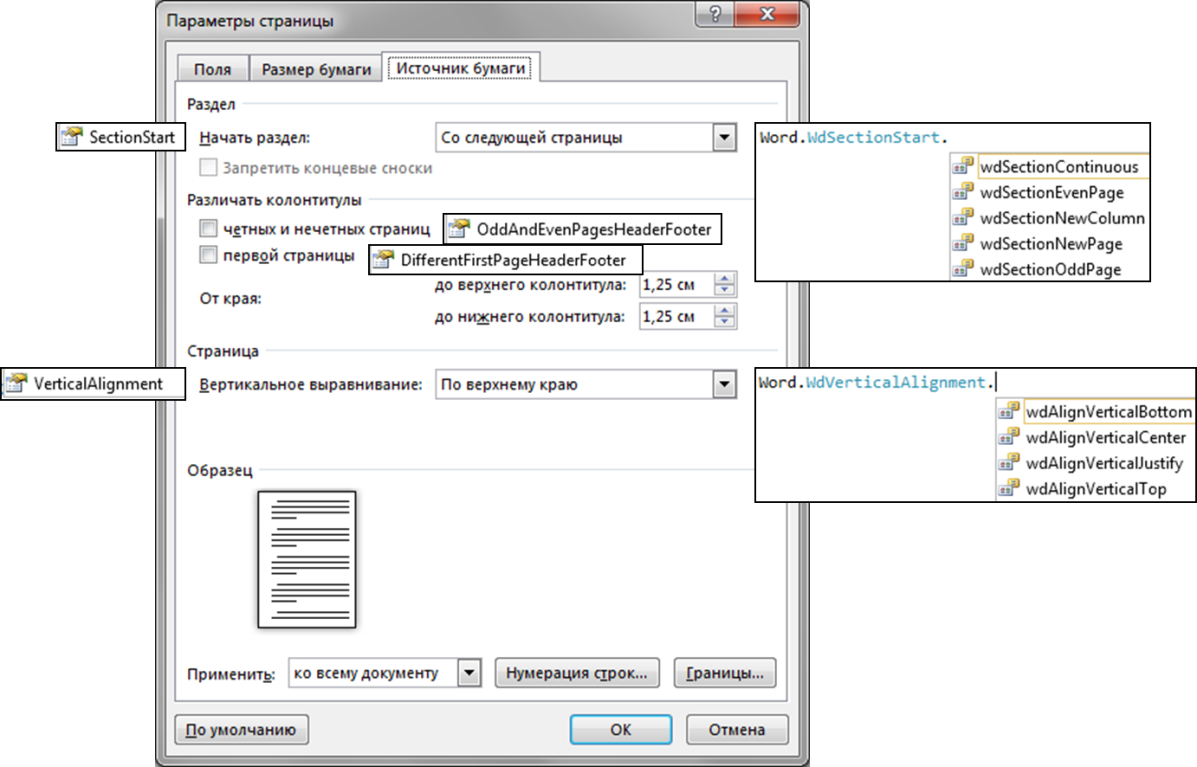


Рисунок 30 – Настройка источников бумаги для документа *Microsoft Office Word* программным способом

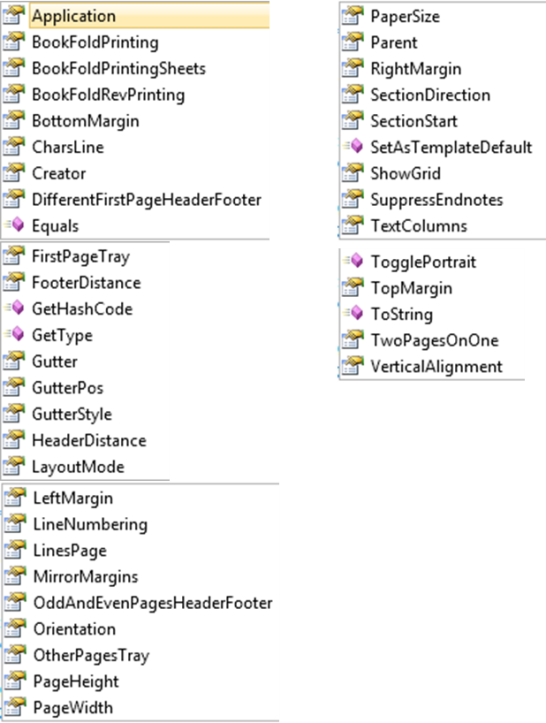


Рисунок 31 – Перечень методов, параметров и свойств документа *Microsoft Office Word* доступных для настройки программным способом в рамках *PageSetup*

**Памятка программисту №7. Пример настройки параметров страницы**

Настройка проводится в контексте блока ***try … catch … finally …*** для возможности отлова непредвиденных ошибок.

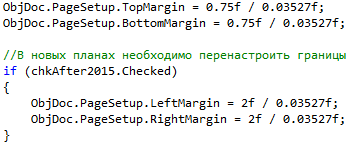


Рисунок 32 – Примерный код, реализующий изменение границ страницы

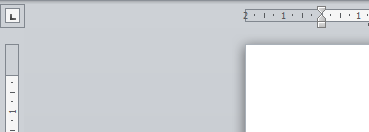


Рисунок 33 – Влияние кодового фрагмента на поведение границ страницы

Альтернативным вариантом записи кода детальной настройки является упаковка в методы. Настройка параметров страницы представлена на Рисунке 34 в контексте отдельного метода.

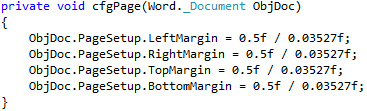


Рисунок 34 – Настройка параметров границ страницы в рамках метода

\* В числителе указан размер границ, заданный в сантиметрах, в знаменателе указана подобранная вручную константа для перевода единиц измерения к интересующему значению. Показано исключительно для понимания. В ***C#*** существует специальный метод для конвертации единиц измерения и его свободно можно использовать в работе.

**Требования к оформлению графического пользовательского интерфейса**:

1. Заголовок главной формы должен содержать надпись вида: «Задание №22 выполнил: [Фамилия И.О. автора]; Номер варианта: [Номер]; Дата выполнения: [дд/мм/гггг]».

2. Дата выполнения проставляется в момент, когда программа считается законченной и по ней можно готовить итоговый отчёт о выполнении работы. Этот момент должен быть согласован с преподавателем, ведущим учебную практику.

3. Имя автора-разработчика графического пользовательского интерфейса вводится через свойства (*Properties*) формы и не должно быть подвержено изменению/редактированию в коде. Это же имя должно быть использовано для вставки в строку Ф.И.О. автора-составителя на титульном листе.

4. Имя и должность лица, проверяющего работу, вводится либо через текстовое поле, либо через комбинированный список, если условием индивидуального задания по варианту не предусмотрено иное.

5. Предусмотреть на графическом пользовательском интерфейсе комбинированный список, в котором выбирается вид документа:

- отчёт;

- реферат;

- эссе;

- курсовой проект;

- курсовая работа;

- доклад;

- домашнее задание.

6. Предусмотреть на графическом пользовательском интерфейсе комбинированный список, в котором выбирается вид работы:

- лабораторная работа;

- практическая работа;

- индивидуальное задание;

- учебная практика;

- производственная практика;

- преддипломная практика.

7. Предусмотреть комбинированный список, позволяющий выбрать номер работы (диапазон от 1 до 10).

8. Предусмотреть текстовое поле для ввода наименования работы.

9. Предусмотреть текстовое поле для ввода темы работы.

10. Предусмотреть текстовое поле для ввода наименования дисциплины, в рамках которой выполняется работа.

11. Между названием вуза и наименованием кафедры должен иметь место один из элементов:

- графический разделитель типа «горизонтальная линия» (номер варианта делится на 3 без остатка);

- нижняя граница таблицы, состоящей из одной строки и одной колонки, растянутой по ширине листа (номер варианта делится на 3 и даёт в остатке единицу);

- заполнитель типа «нижнее подчёркивание», выполненный посредством табуляции (номер варианта делится на 3 и даёт в остатке двойку).

**Варианты индивидуального задания:**

1. Настроить титульный лист без выполнения межстрочных пробелов (пустых абзацев). Для решения задачи необходимо настраивать только абзацный отступ перед и после надписи созданной надписи.

2. Предоставить пользователю возможность выбора города, в котором выполнялась работа, из комбинированного списка. Перечень городов хранить в отдельном текстовом файле. Предусмотреть кнопку «Обновить» для загрузки города, введённого в перечень в процессе работы программы. Исключить необходимость перезапуска программы при добавлении/исключении города из перечня.

3. Выполнить элементы разделов «Выполнил:», «Проверил:» в форме таблицы с невидимыми границами. Предоставить пользователю возможность для настройки левой границы смещения элементов упомянутых разделов.

4. Предоставить пользователю возможность для настройки границ страницы (левое поле, правое поле, нижнее поле, верхнее поле) через дополнительные интерфейсные элементы, размещённые на графическом пользовательском интерфейсе.

5. Предоставить пользователю возможность для настройки текстового интервала (обычный, разреженный, уплотнённый) сразу для всех элементов на титульном листе.

6. Реализовать настройку форматирования титульного листа полностью с использованием таблицы с невидимыми границами.

7. Предоставить пользователю возможность для выгрузки «по галочке» не только титульного листа, но и полноценного шаблона для отчёта по работе с разделами: «Цель работы», «Задача», «Содержательная часть», «Вывод». Каждый раздел должен располагаться на отдельной странице. Использовать для переноса не пустые абзацы, а разрывы страницы.

8. Предоставить пользователю возможность для настройки надписей с именем автора и лица, проверяющего работу по формату шрифта (полужирный, курсив, подчёркивание и др.) через комбинированный список. Настройка индивидуальная.

9. Предоставить пользователю возможность для настройки цвета шрифта для каждого элемента, размещаемого на титульном листе. Настройка индивидуальная.

10. Предоставить пользователю возможность для изменения года выполнения работы в строке «Город – год» посредством нажатия на кнопки «+» и «–» с шагом 1. Левой границей установить -5 лет относительно текущей даты, правой границей установить +5 лет относительно текущей даты.

11. Предоставить пользователю возможность для выбора должности лица, проверяющего работу (ассистент, доцент, профессор) из комбинированного списка, а также ввода в текстовое поле учёной степени лица, проверяющего работу.

12. Установить год поступления в вуз автора-разработчика графического пользовательского интерфейса константой в коде. В зависимости от сочетания параметров текущего года и текущего месяца автоматизировать назначение курса в номере учебной группы.

13. Предоставить пользователю возможность для изменения величины смещения от левого края блоков надписей, связанных с разделами титульного листа «Выполнил:» и «Проверил:». Для каждого раздела настройка смещения индивидуальна.

14. Предоставить пользователю дополнительный комбинированный список, задающий семейство шрифтов (*Calibri*, *Times New Roman*, *Courier New* и др.), влияющий абсолютно на все элементы, размещённые на титульном листе. При необходимости ввести поправки на позиционирование элементов для различных шрифтов.

15. Предоставить пользователю возможность для каждого текстового элемента, размещаемого на титульном листе, активировать форматирование с использованием *CapsLock*, задаваемое посредством логического элемента типа «*CheckBox*».

16. Предоставить пользователю возможность для настройки текста с названием кафедры и названием института через графический пользовательский интерфейс. Настройка указанных надписей индивидуальная. Под детальной настройкой понимается изменение интервала шрифта (обычный / разреженный / уплотнённый), а также размера шрифта с использованием кнопок «+» и «–», регулирующих значение с шагом 2.

17. Предоставить пользователю возможность для детальной настройки текста с наименованием ведомства, регалиями и названием вуза через графический пользовательский интерфейс. Настройка указанных надписей синхронная. Под детальной настройкой понимается изменение семейства и размера шрифта, параметров шрифта (жирный / курсив / подчёркивание), позиционирование текста (по левому краю / по правому краю / по центру / по ширине).