МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение высшего образования  
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК  
Кафедра программного обеспечения

КУРСОВАЯ РАБОТА ПО НАПРАВЛЕНИЮ  
«Математическое обеспечение и администрирование информационных систем»

На тему «Разработка приложения для конструирования экзаменационных билетов»

Выполнил студент 3 курса:  
22МОиАИС 184-1 группы  
Ибраев Ерлан Иржанович

Научный руководитель:  
Павлова Елена Александровна

Оглавление

[Введение 3](#_Toc74213136)

[Глава 1. Теоретическая сведения 4](#_Toc74213137)

[§1.1 Вводные сведения 4](#_Toc74213138)

[§1.2 Формулировка задачи 4](#_Toc74213139)

[§1.3 Метод динамического программирования. 4](#_Toc74213140)

[Глава 2. Реализация приложения 6](#_Toc74213141)

[§2.1 Используемые технологии 6](#_Toc74213142)

[§2.2 Загрузка данных 6](#_Toc74213143)

[§2.3 Алгоритм выборки вопросов 7](#_Toc74213144)

[§2.3 Алгоритм заполнения билета 8](#_Toc74213145)

[Глава 3. Программная реализация 9](#_Toc74213146)

[3.1 Структура данных 9](#_Toc74213147)

[Глава 4. Функционал разработанного приложения и примеры 11](#_Toc74213148)

[4.1. Демонстрация работы программы 11](#_Toc74213149)

[Заключение 14](#_Toc74213150)

[Список литературы 15](#_Toc74213151)

[Приложения 16](#_Toc74213152)

# Введение

Преподаватели сталкиваются с проблемой составления экзаменационных билетов. Весь процесс происходит вручную, на это уходит много времени. Преподавателю нужно учитывать, чтобы билет был уникальный и определенного уровня сложности. Билеты могут включать разное количество заданий: практических задач и теоретических вопросов. Иногда преподаватель может провести экзамен в форме теста из-за этого появляется проблема не повторяющихся вопросов. В качестве решения данной проблемы заключаются в реализации алгоритма на основе «Задача о рюкзаке», который бы распределял вопросы по билетам с учетом сложности и количество уникальных билетов. Такое решение позволит сэкономить время и не нагружать преподавателя.

Проблема данной курсовой является создания уникальных экзаменационных билетов или теста за минимальное количество затраченного времени.

Цель работы – разработать приложения для генерация экзаменационных билетов.

Необходимо решить задачи:

* Изучить библиотеки работы с файлами типа “.docs”
* Необходимо реализовать алгоритмы: алгоритм выборки вопросов, алгоритм загрузки вопросов\задач\тестовые вопросы с вариантом ответа из Word файла, алгоритм заполнение билета по шаблону.
* Реализовать пользовательский интерфейс. В интерфейсе должен быть настройки: тип билета, уровень сложности, редактирования шаблона, количество вопросов и количество билетов.

# Глава 1. Теоретическая сведения

## §1.1 Вводные сведения

Основной инструмент решения поставленной проблемы – это алгоритм выборки вопросов. Содержания текстов вопросов загружается весь и затем на основе этих вопросов разбить на уровни сложности и количество вопросов в билете из этого выходит, что это задача о «рюкзаке».

Задача о рюкзаке, где нам дано N вопросов и ni вопрос имеет сложность wi > 0 и количество pi > 0. Необходимо выбрать из этих вопросов такой набор, чтобы суммарная сложность не превосходила заданной величины W (количество вопросов), а суммарная сложность была максимальна.

## §1.2 Формулировка задачи

Дано N вопросов, W — количество вопросов в билете, w={w1,w2,…,wn} — соответствующий ему набор сложности, p={p1,p2,…,pn} — соответствующий ему набор количеству. Нужно найти набор бинарных величин B={b1,b2,…,bn}, где bi=1 если предмет ni включен в набор, bi=0, если предмет ni не включен, и такой что:

1. b1w1+⋯+bNwN⩽W
2. b1p1+⋯+bN pN максимальна.

Алгоритм сделан на основе динамического программирования. Сложность — O(NW).

## §1.3 Метод динамического программирования.

Пусть A(k,s) есть максимальная количество вопросов, которые можно уложить в рюкзак вместимости s, если можно использовать только первые k вопросов, то есть {n1,n2,…,nk} назовем этот набор допустимых билетов для A(k,s).  
 A(k,0)=0 A(0,s)=0

Найдем A(k,s). Возможны 2 варианта:

1. Если вопрос k не попал в рюкзак. Тогда A(k,s) равно сложность с такой же вместимостью и набором допустимых предметов {n1,n2,…,nk−1}, то есть A(k,s)=A(k−1,s).
2. Если k попал в рюкзак. Тогда A(k,s) равно максимальной сложности, где сложность s уменьшаем на сложность k-ого вопроса и набор допустимых предметов {n1,n2,…,nk−1}{n1,n2,…,nk−1} плюс стоимость kk, то есть A(k−1,s−wk)+pk



То есть: A(k,s)=max(A(k−1,s),A(k−1,s−wk)+pk). Сложность искомого набора равна A(N,W), так как нужно найти максимальную сложность рюкзака, где все вопросы допустимы и вместимость рюкзака W.

Метод динамического программирование всё равно не позволяет решать задачу за полиномиальное время, потому что его сложность зависит от максимального веса. Задача о ранце (или задача о рюкзаке) — одна из NP-полных задач комбинаторной оптимизации.

# Глава 2. Реализация приложения

## §2.1 Используемые технологии

Приложение состоит из несколько модулей: загрузчик, обработчик вопросов, выгрузчик. Программа написана в среде разработки Microsoft Visual Studio 2019 v 16.9.5.

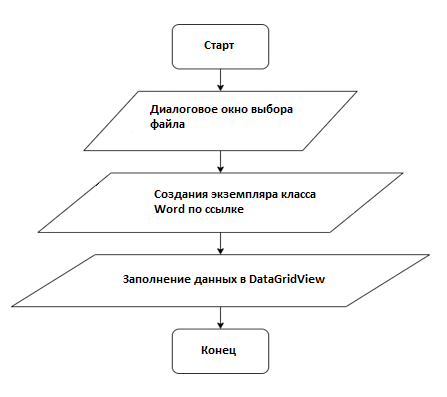
Используемые технологии:

* Язык программирования C#
  + Библиотека Microsoft Interop для разбиения текста по параграфам и текст каждого параграфа выводится в datagridview.
  + WindowsForms - интерфейс программирования приложений.
  + System.Reflection – пространство имен для извлечений сведений.
  + Systerm.IO для чтения и записи в файлы.

## §2.2 Загрузка данных

Дан файл вопросов формата “.docx”. Необходимо добавить вопросы из файла в элемент управления DataGridView.

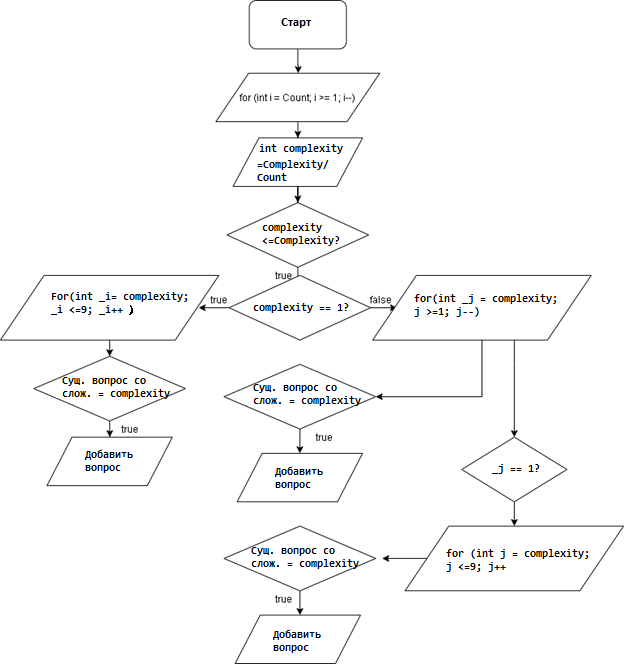
При помощи библиотеки Microsoft Interop создается экземпляр класса Word, которым является наш файл с вопросами. В данном файле текст разбивается по параграфам и текст каждого параграфа выводится в DataGridView.

  
Блок-схема 1. Алгоритм загрузки вопросов/задач из файла

## §2.3 Алгоритм выборки вопросов

Дан элемент управления DataGridView формата билета «вопрос-сложность». Необходимо выбрать вопросы таким образом чтоб сумма их сложностей была равна необходимой.

Все вопросы добавляются в структуру Dictionary<List<question>, int>, где ключом является сложность, а значением лист вопросов с данной сложностью.

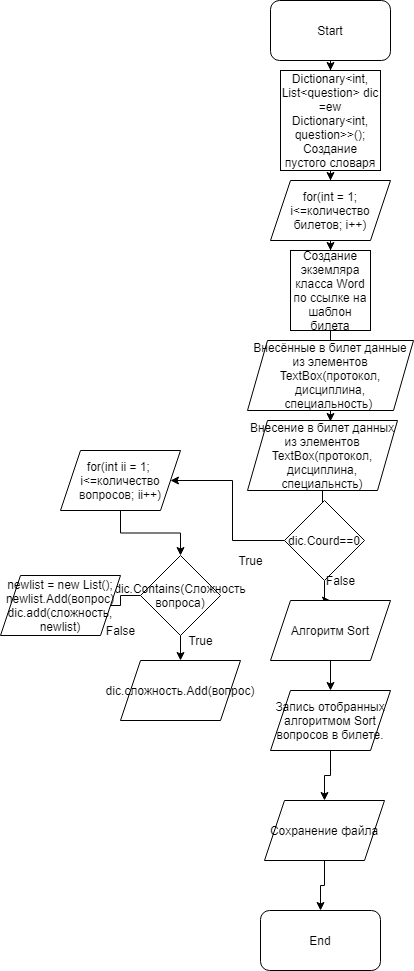
  
Блок-схема 2. Алгоритм Sort

Все вопросы хранятся в коллекцию Dictionary ключом которой является сложность вопроса, а хранимом значением является лист элементами которого являются экземпляра класса Question, при добавлении вопроса в билет данный вопрос удаляется из словаря из Dictionary. Это гарантирует уникальность вопроса в билетах, уникальность сохраняется до тех по мы достаточное количество вопросов для n билетов содержащие k вопросов.

## §2.3 Алгоритм заполнения билета

Дан список вопросов. Необходимо составить билет, заполнив соответствующие места в шаблоне.

Класс BackPack имеет структуру List<questions>, в которой хранятся вопросы, отобранные методом Sort. Метод Method выполняет процесс добавления вопросов в структуру Dictionary<int, List<question>>, с которой работает метод Set, после чего лист вопросов полученных из метода Sort вносится в билет.

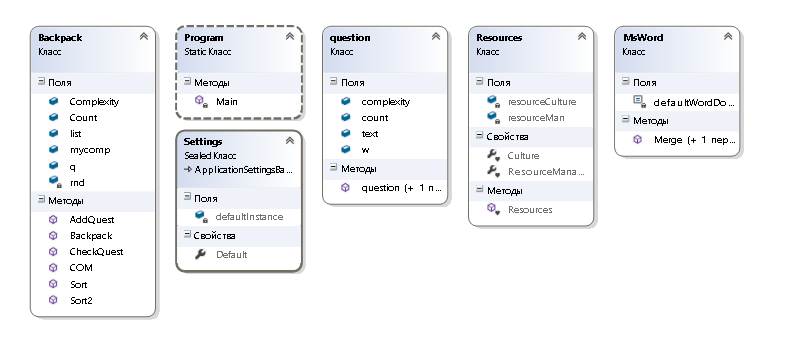
  
Блок-схема 3. Метод п

# Глава 3. Программная реализация

## 3.1 Структура данных

В ходе написания приложения были реализованы классы:

* Question - класс для описания вершин графа.
* BackPack - класс, содержащий методы для работы с графом.
* Form1 - класс, содержащий события для элементов пользовательского интерфейса программы.

  
Диаграмма 1. Иерархия классов

Класс Form1 используется для работы с пользователем, загрузки и сохранения файлов. Основные методы данного класса приведены в таблице 2.

*Методы класса Form1 Таблица 2.*

|  |  |  |
| --- | --- | --- |
| Тип | Название | Описание |
|  | Form1() | Конструктор класса |
|  | button3\_Click | Событие, вызывающее метод Method |
| Button1\_Click | Событие вызывающее диалоговое окно для выбора файла для загрузки вопросов |
| Button2\_Click | Событие, вызывающее диалоговое окно для выбора файла для загрузки задач |
| void | Method() | Метод, реализующий формирование данных из входящих файлов, так же в этом методе вызывается метод Sort |

Класс BackPack представляет собой единичный экземпляр билета. Методы данного класса представлены в таблице 3.

*Методы класса Backpack. Таблица 3.*

|  |  |  |
| --- | --- | --- |
| Тип | Название | Описание |
|  | BackPack(int \_myprice, int \_maxW) | Конструктор класса, билет с ограничением на количество вопросов и суммарной сложностью этих вопросов, где \_myprice – требуемая сложность билета, \_maxW – требуемое количество вопросов |
| int | COM() | Возвращает суммарную сложность вопросов в билете |
| void | Sort(ref Dictionary<int, List<question>> dic) | Метод выборки вопросов в билет, где dic – текущий список вопросов |
| Sort2(ref Dictionary<int, List<question>> dic) | Метод выборки задач в билет, где dic – текущий список задач |
| AddQuest(ref Dictionary<int, List<question>> dic, int median) | Метод, добавляющий экземпляр класса question из dic в List list |
| bool | CheckQuest(int median, Dictionary<int, List<question>> dic) | Метод проверки содержания вопроса в билете, где median = complexity(сложность билета)/Count(кол-во вопросов), dic – словарь хранящий текущий список вопросов |

# Глава 4. Функционал разработанного приложения и примеры

## 4.1. Демонстрация работы программы

При запуске программы перед нами появляется окно шаблона. Он будет показывать, как будет выглядеть экзаменационный билет или тест после генерации

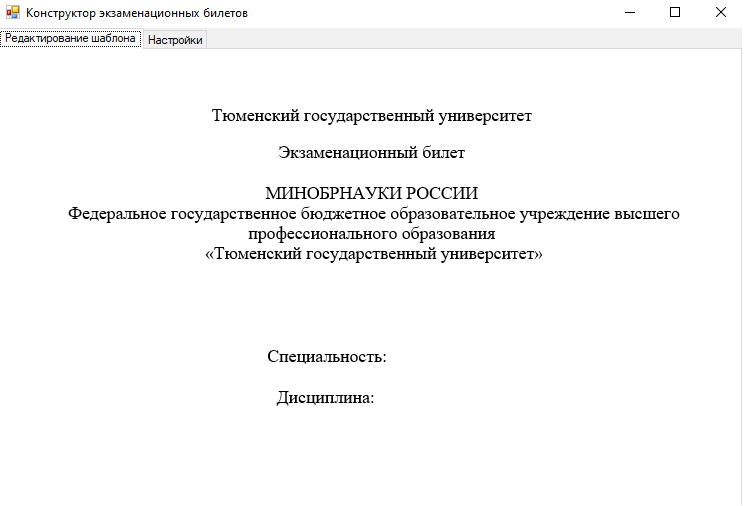
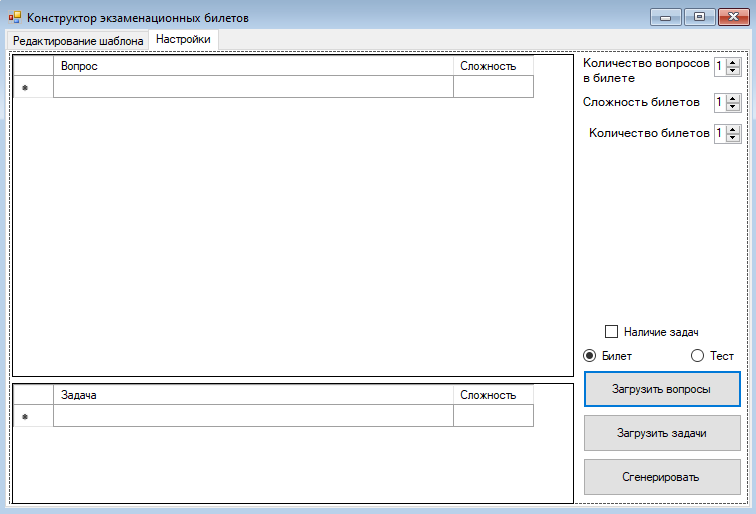


Рис. 1. Окно шаблона.

На следующей вкладке элемента находится окно настройки генерации билетов.

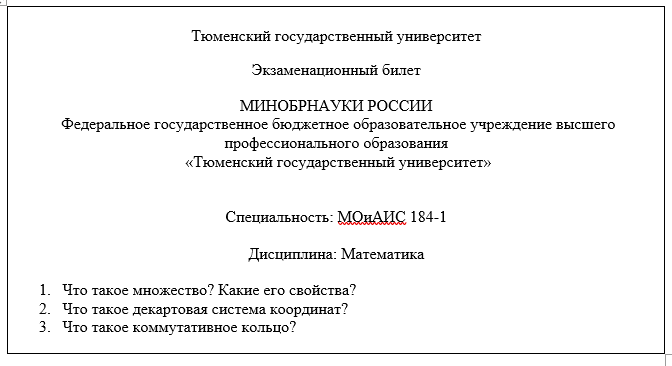
Разработанное приложение обладает следующим функционалом: генерация билетов по заданным настройкам (количество билетов, сложность билетов, количество вопросов в билете)

Нажав на кнопку «Загрузить вопросы» и выбрал в диалоговом окне нужный. docs файл. Все вопросы из выбранного файла мы увидим в соответствующем элементе DataGridView. Если вы хотите, чтобы в экзаменационном билете был практические задачи вам нужно нажать на кнопку «Загрузить задачи» и задать им уровень сложности. При генерации вопросы и задачи будут с комбинированы.  
 Также вы можете задать количество вопросов в билете и уровень сложности и количество билетов.

  
Рис 2. Типы формата экзаменационного билета.

После выполнения метода Set мы будем иметь файл со всеми сгенерированными уникальными, билетами по выбранным параметрам.

На выходе будет получен Word файл следующего вида.

 Рис 3. Пример полученного решения.

# Заключение

В результате работы, с помощью библиотеки Microsoft Interop, был разработан загрузчик данных с вопросами. Разработан алгоритм выборки вопросов для составления уникального билета или теста, с помощью элемента управления DataGridView «вопрос-сложность» или «вопрос-ответ-сложность». Разработан алгоритм для выгрузки экзаменационного билета или теста с шаблоном. Разработка пользовательского приложения, предоставляющее пользователю доступ конфигурации к билетам и работы с ними.

# Список литературы

1. Bestprog [Электронный ресурс]. – режим доступа [www.bestprog.net](http://www.bestprog.net) – элементы управления DataGridView (Дата обращения 29.05.2021)
2. Microsoft.docs [Электронный ресурс]. – режим доступа www.docs.microsoft.com – DataGridView (Дата обращения 29.05.2021)
3. Code-Project [Электронный ресурс]. – Режим доступа: [www.codeproject.com](http://www.codeproject.com) – Genetic algorithms for knapsack problem. (Дата обращения 01.06.2021).
4. Knapsack problem[Электронный ресурс]. – Режим доступа: www.geeksforgeeks.org/ –0-1Knapsack. (Дата обращения 01.06.2021).
5. Stack Overflow [Электронный ресурс]. – Режим доступа: <https://stackoverflow.com/> – Stack Overflow – online community for programmers. (Дата обращения 03.06.2021).
6. Word Documentation [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com /en/ – Word – Documentation. (Дата обращения 04.06.2021).
7. В. А. Касторнова. «Структуры и алгоритмы компьютерной обработки данных». БХВ-Петербург, 2019. – 672 с.

# Приложения

Программа реализации класса BackPack

class Backpack  
 {  
 public int Count;   
 public Dictionary<int, List<question>> q = new Dictionary<int, List<question>>();  
 public Dictionary<int, List<question>, List<answer>> q = new Dictionary<int, List<question>, List<answer>();  
 public List<question> list = new List<question>();  
 public int Complexity;  
 public Backpack(int \_myprice, int \_maxW)  
 {   
 Count = \_maxW;  
 Complexity = \_myprice;  
 }  
 public bool CheckQuest(int median, Dictionary<int, List<question>> dic)  
 {  
 if(dic.ContainsKey(median) && dic[median].Count > 0)  
 {  
 return true;  
 }  
 else  
 {  
 return false;  
 }  
 }  
 Random rnd = new Random();  
 public void AddQuest(ref Dictionary<int, List<question>> dic, int median)  
 {  
 int g = rnd.Next(0, dic[median].Count);  
 list.Add(dic[median][g]);  
 Complexity -= dic[median][g].complexity;  
 dic[median].Remove(dic[median][g]);  
 if(dic[median].Count == 0)  
 {  
 dic.Remove(median);  
 }   
 }   
 public void Sort(ref Dictionary<int, List<question>> dic)  
 {  
 int median = 0;   
 for (int i = Count; i >= 1; i--)  
 {  
 median = Complexity / i;  
 if (median <= Complexity)  
 {  
 if (median == 1)  
 {  
 for (int \_i = median; \_i <= 9; \_i++)  
 {  
 if (CheckQuest(\_i,dic))  
 {  
 AddQuest(ref dic, \_i);  
 break;  
 }  
 }  
 }  
 else  
 {  
 for (int j = median; j >= 1; j--)  
 {  
 if (CheckQuest(j,dic))  
 {  
 AddQuest(ref dic, j);  
 break;  
 }   
 if(j==1)  
 {  
 for (int \_j = median; \_j <= 9; \_j++)  
 {  
 if (CheckQuest(\_j,dic))  
 {  
 AddQuest(ref dic, \_j);  
 break;  
 }  
 }  
 }  
  
 }   
 }  
 }   
 }  
 int p = 1;  
 if (COM() - median <= 1)  
 {  
 Sort2(ref q, median);  
 }  
 else  
 {  
 Sort2(ref q, 1);  
 }  
 }  
 public int COM()  
 {  
 int c = 0;  
 for (int i = 0; i < list.Count; i++)  
 {  
 c += list[i].complexity;  
 }  
 return c;  
 }  
 public void Sort2(ref Dictionary<int, List<question>> dic, int Count)  
 {  
 Complexity = Count;  
 for (int i = Count; i >= 1; i--)  
 {  
 int median = Complexity / i;  
 if (median <= Complexity)  
 {  
 if (median == 1)  
 {  
 for (int \_i = median; \_i <= 9; \_i++)  
 {

if (CheckQuest(\_i, dic))  
 {  
 AddQuest(ref dic, \_i);  
 return;  
 }  
 }  
 }  
 else  
 {  
 for (int j = median; j >= 1; j--)  
 {  
 if (CheckQuest(j, dic))  
 {  
 AddQuest(ref dic, j);  
 break;  
 }  
 if (j == 1)  
 {  
 for (int \_j = median; \_j <= 9; \_j++)  
 {  
 if (CheckQuest(\_j, dic))  
 {  
 AddQuest(ref dic, \_j);  
 break;  
 }  
 }  
 }  
 }  
 }  
 }  
 }   
 }

Алгоритм выгрузки

public void Set()  
 {  
 Dictionary<int, List<question>> dic = new Dictionary<int, List<question>>();  
 Dictionary<int, List<question>> dic2 = new Dictionary<int, List<question>>();  
 Dictionary<int, List<question>> dic3 = new Dictionary<int, List<question>, List<answer >>();  
 Dictionary<int, List<question>> dic4 = new Dictionary<int, List<question>. List<answer >>();  
 Backpack bp = new Backpack((int)numericUpDown2.Value, (int)numericUpDown1.Value);  
 for (int i = 1; i <= numericUpDown3.Value; i++)  
 {  
 this.Text = "Processing:" + Convert.ToString(100 \* i / (int)numericUpDown3.Value);  
 Object missingObj = Missing.Value;  
 Object trueObj = true;  
 Object falseObj = false;  
 object file = @"C:\Users\Gaukhar Kanapyanova\Desktop\Курсовая работа 4 семестр\Курсовая работа 4 семестр\bin\Debug\shablon.doc";  
 app = new Word.Application();  
 doc = app.Documents.Add(ref file);  
 Word.Range r = doc.Range();  
 t = doc.Tables[1];  
 t.Range.Paragraphs[14].Range.Text = textBox1.Text + "\n";  
 t.Range.Paragraphs[14].Range.Text = textBox2.Text + "\n";  
 t.Range.Paragraphs[17].Range.Text = textBox3.Text + "\n";  
 t.Range.Paragraphs[19].Range.Text = $"Экзаменационной билет №{i}\n";  
 t.Range.Paragraphs[19].Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;  
 bp = new Backpack((int)numericUpDown2.Value, (int)numericUpDown1.Value);  
 Random rnd = new Random();  
 if (dic.Count == 0 || dic.Values.Count == 3)  
 {  
 dic = new Dictionary<int, List<question>>();  
 int g = dataGridView1.Rows.Count - 1;  
 for (int ii = 0; ii < g; ii++)  
 {   
 if (dic.ContainsKey(Convert.ToInt32(dataGridView1[1, ii].Value)))  
 {  
 dic[Convert.ToInt32(dataGridView1[1, ii].Value)].Add(new question(dataGridView1[0, ii].Value.ToString(), Convert.ToInt32(dataGridView1[1, ii].Value), 1));  
 }  
 else  
 {  
 List<question> newlist = new List<question>();  
 newlist.Add(new question(dataGridView1[0, ii].Value.ToString(), Convert.ToInt32(dataGridView1[1, ii].Value), 1));  
 dic.Add(Convert.ToInt32(dataGridView1[1, ii].Value), newlist);  
 }  
 }  
 }  
 if (checkBox1.Checked && dic2.Values.Count == 0)  
 {  
 dic2 = new Dictionary<int, List<question>>();  
 for (int ii = 0; ii < dataGridView2.Rows.Count - 1; ++ii)  
 {  
 if (dic2.ContainsKey(Convert.ToInt32(dataGridView2[1, ii].Value)))  
 {  
 dic2[Convert.ToInt32(dataGridView2[1, ii].Value)].Add(new question(dataGridView2[0, ii].Value.ToString(), Convert.ToInt32(dataGridView2[1, ii].Value), 1, 1));  
 }  
 else  
 {  
 List<question> newlist = new List<question>();  
 newlist.Add(new question(dataGridView2[0, ii].Value.ToString(), Convert.ToInt32(dataGridView2[1, ii].Value), 1, 1));  
 dic2.Add(Convert.ToInt32(dataGridView2[1, ii].Value), newlist);  
 }  
 }  
 }  
 bp.q = dic2;  
 bp.Count = (int)numericUpDown1.Value;  
 bp.Complexity = (int)numericUpDown2.Value;  
 bp.Sort(ref dic);   
 for (int j = 0; j < bp.list.Count; j++)  
 {  
 t.Range.Paragraphs[21 + j].Range.Text = $"{j + 1}. " + bp.list[j].text;

t.Range.Paragraphs[21 + j].Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;  
 t.Range.Paragraphs[21 + j].CharacterUnitLeftIndent = 2f;  
 t.Range.InsertParagraphAfter();  
 }  
 doc.SaveAs($@"C:\Users\Erlan\Desktop\Курсовая работа 4 семестр\all\Билет-{i}.docx");  
 doc.Close();  
 app.Quit();  
 app = null;  
 doc = null;  
 }  
 if (checkBox3.Checked)  
 {  
 string[] filePaths = Directory.GetFiles(@"C:\Users\Gaukhar Kanapyanova\Desktop\Курсовая работа 4 семестр\all", "\*.docx", SearchOption.AllDirectories);  
 string[] documentsToMerge = filePaths;  
 string outputPath = String.Format(@"C:\Users\Gaukhar Kanapyanova\Desktop\Билеты.docx");  
 MsWord.Merge(documentsToMerge, outputPath, true, @"C:\Users\Gaukhar Kanapyanova\Desktop\shablon.doc");  
 MessageBox.Show("Файл был сохранен на рабочем столе");  
 try  
 {  
 foreach (string files in documentsToMerge)  
 {  
 File.Delete(files);  
 }  
 }  
 catch  
 { }  
 }  
 this.Text = "Конструктор экзаменационных билетов";  
 }