Міністерство освіти і науки України

Одеський національний університет імені І.І. Мечникова

Інститут математики економіки та механіки

Кафедра теоретичної механіки

**Звіт**

**з навчально-обчислювальної практики**

***«Розробка програмного забезпечення ПЕОМ»***

Виконала:

Студентка 2-го курсу

Напряму «Механіка»

Шокотько Дар’ї

м. Одеса – 2017р.

**Содержание**

1. Постановка задания
2. Написание програмного кода
   1. Окно приветствия
   2. Окно преподавательского режима
   3. Окно студенческого режима
   4. Окно с заданиями билета
3. Результат
4. Приложение (код программы)

**Постановка задания**

**Вход в приложение**

Создать **WPF**-приложение, которое позволяет организовать тестовый прием экзамена. Приложение может работать в двух режимах: режим преподавателя и режим студента. В момент запуска приложение может иметь вид:

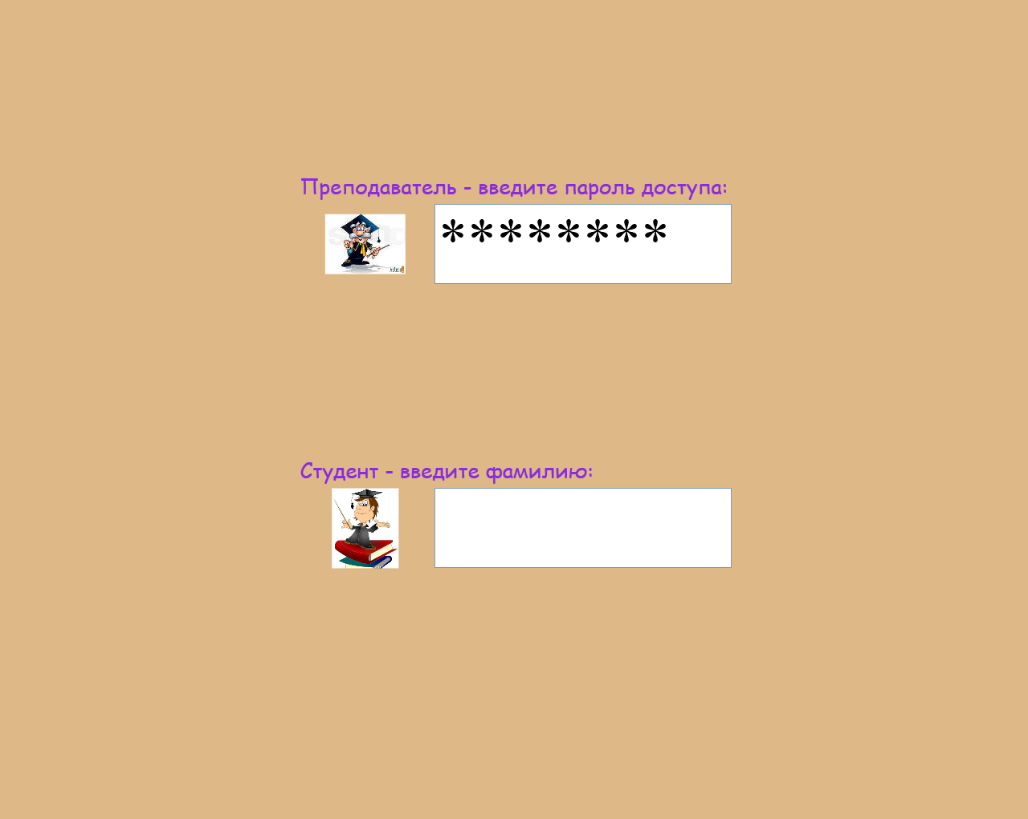


Рис. 1

Окно разворачивается на весь экран, не имеет строки заголовка. Пользователь может зайти в одном из предложенных режимов. При входе в режим преподавателя необходимо ввести правильный пароль (используется компонента для ввода пароля). Если осуществлен ввод не правильного пароля, то приложение завершает свою работу. При входе в режим студента необходимо ввести фамилию (используется текстовое окно). Фамилией студента является любая не пустая строка, которая будет в дальнейшем использоваться для идентификации студента в процессе проверки его ответов. Фамилия студента должна быть записана в файл с фамилиями, который имеет специальное разрешение.

Вход в приложение осуществляется при нажатии пользователем клавиши **Enter**, когда фокус находит в окне или ввода пароля, или ввода фамилии.

**Режим студента**

При входе в режим студента выдается сообщение с введенной фамилией

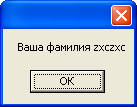


Рис. 2

и открывается форма вида



Рис. 3

Оформление окна приложения зависит от варианта задания. В нижнем правом углу располагается меню вида:

|  |  |
| --- | --- |
| **Выбор темы** | **Вопрос** |
| Тема 1 | Выбор |
| Тема 2 |  |
| Тема 3 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Выход |

Тем для выбора должно быть три, что означает - будут содержаться билеты с тремя разными расширениями (для каждой темы файлы билетов должны иметь свое расширение). Выбранная тема указывается в меню «галочкой». По умолчанию должна быть выбрана первая тема. При нажатии пункта меню **Выбор** открывается диалоговое окно открытия файлов, в котором должно быть не менее трех билетов по указанной теме. При чем должны открываться только файлы с расширением соответствующим данной темы. Например

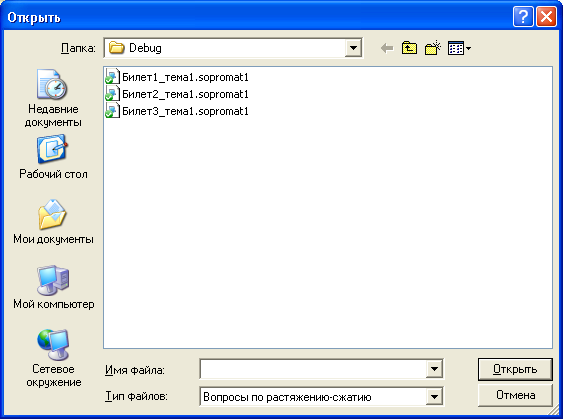


Рис. 4

После выбора открывается форма вида:



Рис. 5

На форме содержатся три вопроса по указанной теме. Вопросы изначально скрыты, при открытии форма принимает вид:

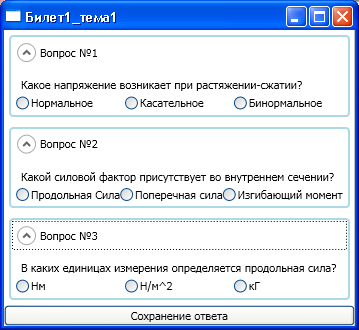


Рис. 6

Пользователь должен указать правильные ответы и сохранить билет. Билет сохраняется под именем вида: zxczxc\_Билет1\_тема1.otvet – фамилия студента\_номер+билета\_тема.otvet. Таким образом, если студент даст другой ответ на этот же билет, то будет сохранен новый ответ.

**Режим преподавателя.**

Для каждой темы, входящий в тесты, необходимо подготовить текстовый файл с вопросами и тремя возможными ответами (можно подготовить в любом редакторе). Количество вопросов должно превышать максимальное количество билетов\*3.

При входе в режим преподавателя открывается форма вида:

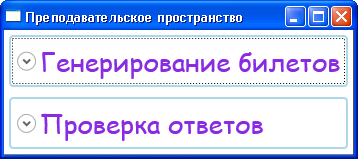


Рис. 7

При открытии области ***Генерирование билетов*** имеем:

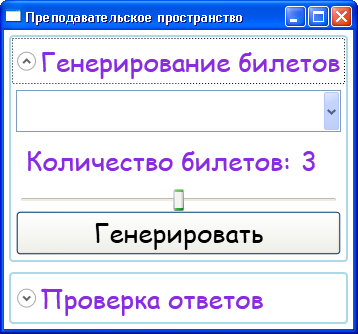


Рис. 8

Информация о количестве билетов (в виде числа) задается с помощью свойства привязки. Выпадающий список содержит список тем. Преподаватель должен выбрать одну из тем:

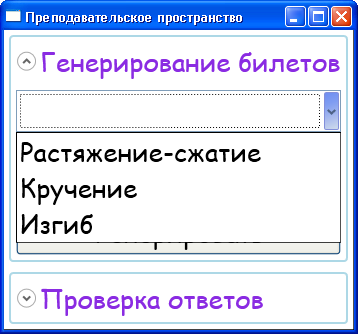


Рис. 9

Задать нужное количество билетов и нажать кнопку ***Генерировать***. Из подготовленного файла случайным образом формируются заданное количество билетов-файлов с именами, как показано на рис. 4.

При открытии области ***Проверка ответов***, формируется список студентов из файла с фамилиями.

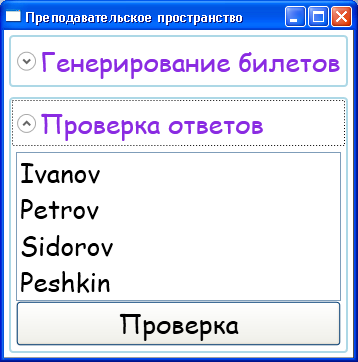


Рис. 10

Преподаватель выбирает фамилию из списка и нажимает кнопку ***Проверить***. С помощью класса **DirectoryInfo** (лекции 1–го курса) организовать просмотр содержимого папки с ответами и проверить все сохраненные ответы выбранного студента. Оценка студента заносится в файл с оценками.

**Написание программного кода**

**Окно приветствия**

Окно открывается на весь экран без рамок, с заданным цветом. Оно содержит в себе два *TextBlock*’а, два изображения и два поля ввода (одно из которых имеет тип Password). Объекты распределены по форме с помощью контейнеров *Grid* и *StackPanel*.

Для обработки нажатия клавиши Enter на полях ввода используются события *KeyDown* и соответствующие событийные функции: *pwdBox\_KeyDown* для поля ввода пароля и *studBox\_KeyDown* для поля ввода фамилии студента.

Функция *pwdBox\_KeyDown* сравнивает введенный пароль с паролем преподавателя и, если они совпадают – открывает окно преподавательского режима, закрывая текущее окно.

Функция *studBox\_KeyDown* проверяет не была ли введена пустая строка и после вывода фамилии студента с помощью *MessageBox*’а, открывает окно студенческого режима (передавая в конструктор строку с фамилией студента), закрывая текущее окно.

Если пароль преподавателя был введен неверно, или не была введена фамилия студента – окно закрывается.

**Окно преподавательского режима**

В окне расположены 2 эспандера, которые при открытии окна свернуты. При их разворачивании окно автоматически растягивается под размеры их содержимого (это достигается с помощью свойства *SizeToContent="Height"*).

У эспандеров установлено определенное семейство шрифта и стиля написания, которые наследуются всеми элементами, содержащимися в них

В первом эспандере содержаться два лейбла, выпадающий список, кнопка и слайдер, скомпонованные с помощью *StackPanel*.

Один из лейблов содержит свойство привязки к слайдеру для отображения количества генерируемых билетов. У слайдера выставлены свойства *TickFrequency="1"* и *IsSnapToTickEnabled="True"*, для перемещения только по целым числам, а также минимальное (0) и максимальное (8) значения. При открытии окна, ползунок слайдера находится на отметке *3*.

При нажатии кнопки «Генерировать» вызывается событийная функция *Button\_Click\_Gen*, которая передает переменной *path* путь файла с вопросами выбранной темы (если тема не выбрана, появляется соответствующее сообщение с просьбой выбрать тему для генерации). Далее, с помощью класса *DirectoryInfo* проверяется количество билетов по выбранной теме. Путь к файлу с заранее составленными вопросами, номер темы и номер билета передаются в функцию *genCard* в цикле for по количеству генерируемых билетов. Данная функция считывает файл при помощи класса *StreamReader* и записывает его построчно в одномерный массив типа string. Для того, чтобы вопросы в билете не повторялись, создается рандомный целочисленный массив, длина которого соответствует количеству вопросов в билете, массив сортируется в порядке возрастания его элементов. Затем в двумерный строковый массив записываются вопрос и варианты ответов к нему, соответственно целочисленному массиву, определенному ранее. Потом с помощью класса *StreamWriter*, создается и записывается файл с билетом:

*«Cards/Topic{номер темы}\_card{номер билета}.Topic{номер темы}»*

Второй эспандер содержит в себе список элементов и кнопку «Проверить». При загрузке окна, в список элементов записываются фамилии студентов, которые сохранили ответ хотя бы на один билет, из соответствующего файла. При нажатии на кнопку «Проверить», вызывается событийная функция *Button\_Click\_Prov*, которая из файлов с ответами на билеты, название которых содержит фамилию выбранного студента, считывает количество правильных ответов (правильный ответь отмечен символом ‘$’) и выдает сообщение с количеством баллов набранных студентом (один правильный ответ - 3 балла), а также записывает эту информацию в соответствующий файл со всеми оценками.

Если, при на нажатии на кнопку «Проверить», не была выбрана фамилия студента из списка, то приложение выдает соответствующее сообщение, с просьбой выбрать студента.

**Окно студенческого режима**

Окно студенческого режима содержит меню выбора темы, закрепленное с помощью *DockPanel* в середине окна. В качестве фона выбрана картинка, заданная с помощью свойства *Window.Background*.

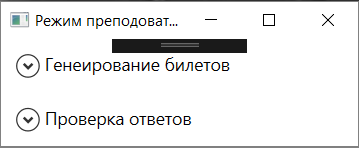
После выбора пользователем темы из меню, он выбирает билет, нажав на кнопку «Выбор билета». При нажатии вызывается событийная функция *Choose\_Click*, в которой выполняется проверка выбранной темы (по умолчанию – Тема1), в зависимости от которой создается фильтр для *OpenFileDialog*: «*Тема {номер темы} (\*.Topic{номер темы})|\*.Topic{номер темы}*».

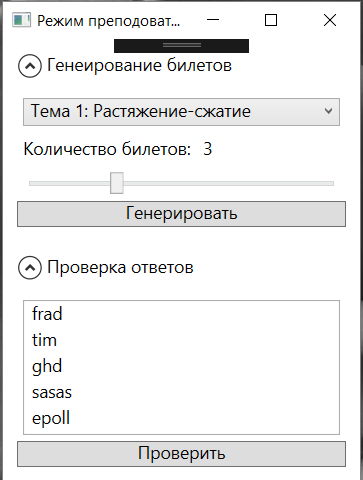
После выбора билета отрывается окно *AnswerForm*, в конструктор которой передается путь файла билета и фамилия студента. При открытии этого окна, с помощью функции *SetAll*, вызываемой при инициализации *AnswerForm*, считываются вопросы и варианты ответов из билета и выводятся в соответствующие лейблы и радиокнопки, содержащиеся в эспандерах, соответствующих номерам вопросов.

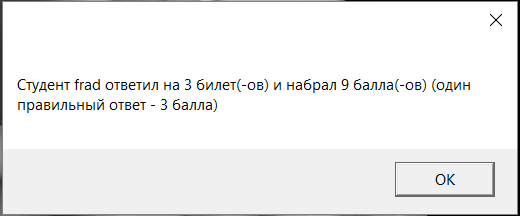
Эспандеры и кнопка «Сохранить» скомпонованы в окне при помощи *StackPanel*. После того, как студент, выбрав ответы, нажимает «Сохранить», вызывается событийная функция *Button\_Click*, обеспечивающая запись ответов в соответствующий файл: «*Answers/{фамилия студента}\_ Topic{номер темы}\_card{номер билета}.otvet*», и окно закрывается.

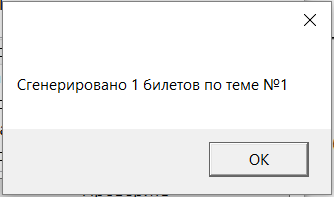
**Результат**

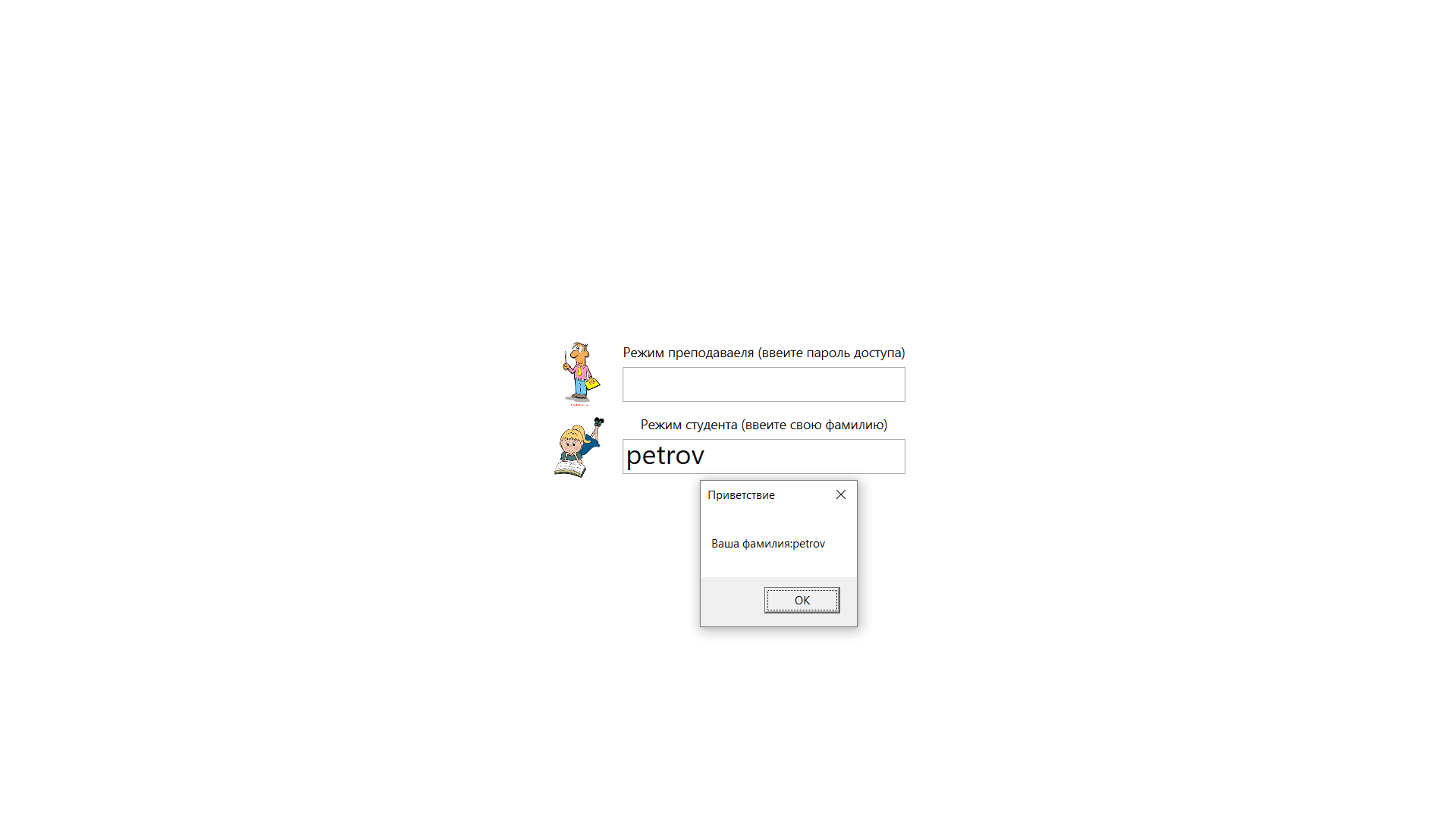


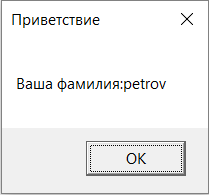




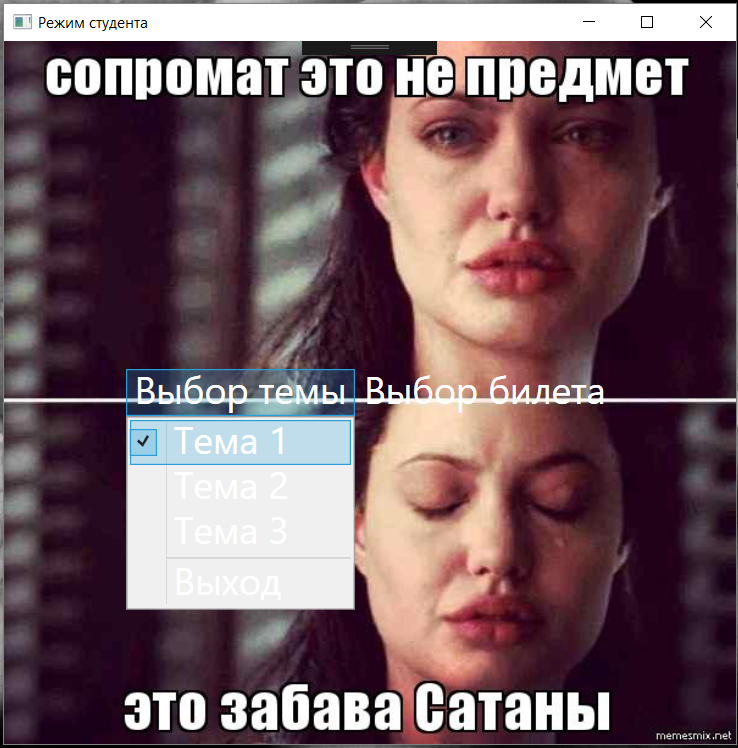


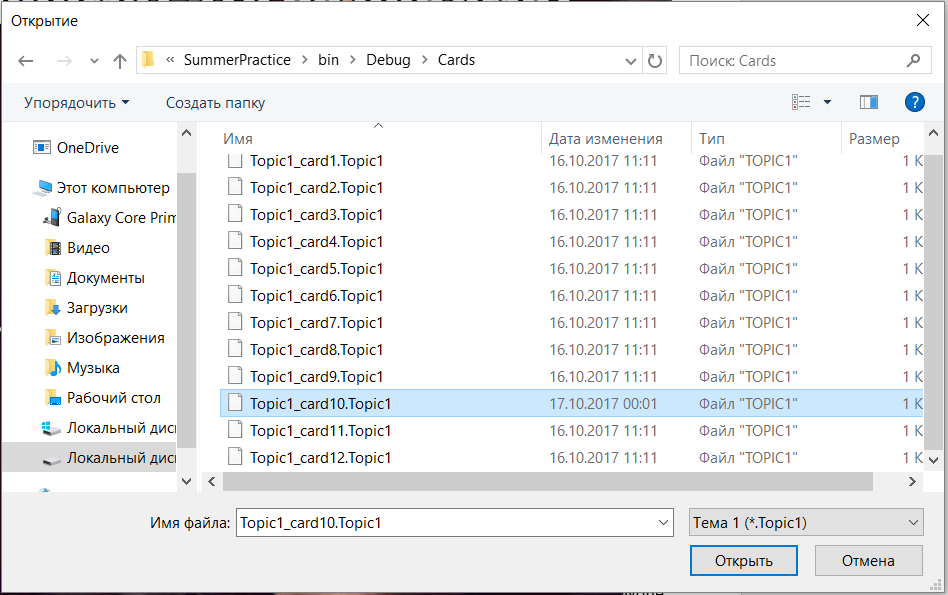


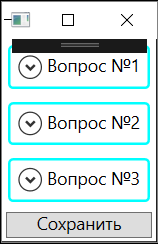


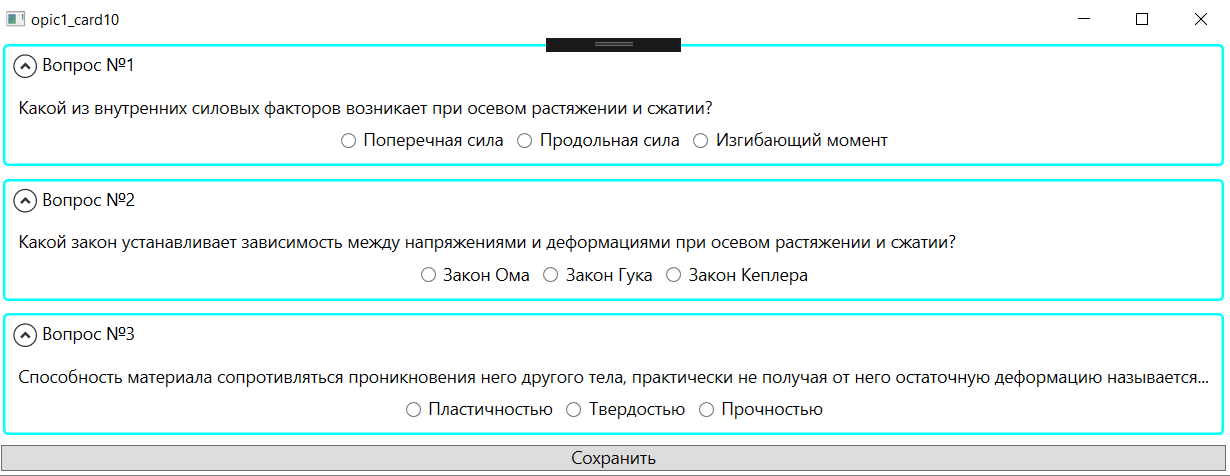












**Приложение (код программы)**

MainWindow.xaml

<Window x:Class="SummerPractice.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:SummerPractice"

mc:Ignorable="d"

Title="SturtWindow" Height="350" Width="525" WindowStyle="None" WindowState="Maximized">

<Grid>

<Grid.RowDefinitions>

<RowDefinition ></RowDefinition>

<RowDefinition Height="auto"></RowDefinition>

<RowDefinition Height="auto"></RowDefinition>

<RowDefinition ></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition ></ColumnDefinition>

<ColumnDefinition Width="auto"></ColumnDefinition>

<ColumnDefinition Width="auto"></ColumnDefinition>

<ColumnDefinition ></ColumnDefinition>

</Grid.ColumnDefinitions>

<Image Grid.Row="1" Grid.Column="1" Source="Images/lect.jpg" Width="70" Margin="3"></Image>

<Image Grid.Row="2" Grid.Column="1" Source="Images/stud.jpg" Width="70" Margin="3"></Image>

<StackPanel Grid.Row="1" Grid.Column="2" Margin="5">

<TextBlock Margin="3" FontFamily="Copperplate Gothic Light" FontSize= "14" TextAlignment="Center">

Режим преподаваеля (ввеите пароль доступа)

</TextBlock>

<PasswordBox Margin="3" Name="pwdBox" MaxLength="64" PasswordChar="\*" MinHeight="35" FontFamily="Copperplate Gothic Light" FontSize= "29" KeyDown="pwdBox\_KeyDown"></PasswordBox>

</StackPanel>

<StackPanel Grid.Row="2" Grid.Column="2" Margin="5">

<TextBlock Margin="3" FontFamily="Copperplate Gothic Light" FontSize= "14" TextAlignment="Center">

Режим студента (ввеите свою фамилию)

</TextBlock>

<TextBox MinHeight="35" Margin="3" Name="studBox" FontFamily="Copperplate Gothic Light" FontSize= "29" KeyDown="studBox\_KeyDown"/>

</StackPanel>

<!--<Button Grid.Column="3" Grid.Row="3" Margin="3" FontFamily="Copperplate Gothic Light" FontSize= "14" Height="20" Click="Button\_Click" Background="White" BorderBrush="White" Width="50">

Выход

</Button>-->

</Grid>

</Window>

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace SummerPractice

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void pwdBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

if (pwdBox.Password == "123456")

{

LectureForm t = new LectureForm();

t.Show();

this.Close();

}

else

{

this.Close();

}

}

}

private void studBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

if (studBox.Text.Length > 1)

{

MessageBox.Show("Ваша фамилия:" + studBox.Text, "Приветствие", MessageBoxButton.OK);

StudentForm m = new StudentForm(studBox.Text);

m.Show();

this.Close();

}

else

{

this.Close();

}

}

}

}

}

LectureForm.xaml

<Window x:Class="SummerPractice.LectureForm"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:SummerPractice"

mc:Ignorable="d"

Title="Режим преподователя" SizeToContent="Height" Width="300" WindowStartupLocation="CenterScreen">

<StackPanel>

<Expander Margin="5" Padding="5" Header="Генеирование билетов" FontFamily="Copperplate Gothic Light" FontSize= "14" >

<StackPanel>

<ComboBox Name="top" Margin="5" IsReadOnly="True" >

<ComboBoxItem>Тема 1: Растяжение-сжатие</ComboBoxItem>

<ComboBoxItem>Тема 2: Кручение</ComboBoxItem>

<ComboBoxItem>Тема 3: Изгиб</ComboBoxItem>

</ComboBox>

<StackPanel Orientation="Horizontal">

<Label>Количество билетов:</Label>

<Label Content="{Binding ElementName=bil, Path=Value}"></Label>

</StackPanel>

<Slider TickFrequency="1" IsSnapToTickEnabled="True" Name="bil" Margin="5" Maximum="8" Minimum="1" Value="3"></Slider>

<Button Content="Генерировать" Click="Button\_Click\_Gen"></Button>

</StackPanel>

</Expander>

<Expander Margin="5" Padding="5" Header="Проверка ответов" FontFamily="Copperplate Gothic Light" FontSize= "14">

<StackPanel>

<ListBox Name="ListBox" Margin="5"></ListBox>

<Button Content="Проверить" Click="Button\_Click\_Prov" ></Button>

</StackPanel>

</Expander>

</StackPanel>

</Window>

LectureForm.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.IO;

namespace SummerPractice

{

/// <summary>

/// Interaction logic for LectureForm.xaml

/// </summary>

public partial class LectureForm : Window

{

string path;

public LectureForm()

{

InitializeComponent();

string names;

StreamReader sr = new StreamReader("Students.txt");

names = sr.ReadToEnd();

sr.Close();

string[] mas = names.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);

for (int i = 0; i < mas.Length; i++)

{

ListBox.Items.Add(mas[i]);

}

}

private void Button\_Click\_Gen(object sender, RoutedEventArgs e)

{

if (top.SelectedIndex == -1)

{

MessageBox.Show("Выберите тему!!!", "", MessageBoxButton.OK);

return;

}

path = "Topics/Topic" + Convert.ToString(top.SelectedIndex + 1) + ".txt";

DirectoryInfo di = new DirectoryInfo("Cards/");

int kol = 0;

foreach (var fi in di.GetFiles(String.Format("Topic{0}\_card?.Topic{0}", top.SelectedIndex + 1)))

{

kol++;

}

for (int i = 1; i <= Convert.ToInt32(bil.Value); i++)

{

genCard(path, top.SelectedIndex + 1, i + kol);

}

MessageBox.Show(String.Format("Сгенерировано {0} билетов по теме №{1}", Convert.ToInt32(bil.Value), top.SelectedIndex + 1), "", MessageBoxButton.OK);

}

private void genCard(string p, int top, int q)

{

Encoding enc = Encoding.GetEncoding(1251);

StreamReader sr = new StreamReader(p, enc);

int count = 0;

while (!sr.EndOfStream)

{

sr.ReadLine();

count++;

}

sr.Close();

int[] mas = new int[3];

Random r = new Random();

//чтоб не повторялись

for (int i = 0; i < mas.Length; i++)

{

mas[i] = r.Next(count / 4);

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

{

if (mas[j] == mas[i])

{

i--;

}

}

}

Array.Sort(mas);

string[] strmas = new string[count];

StreamReader sr1 = new StreamReader(p, enc);

count = 0;

while (!sr1.EndOfStream)

{

strmas[count] = sr1.ReadLine();

count++;

}

sr1.Close();

string[,] finalmas = new string[3, 4];

for (int j = 0; j < 3; j++)

{

finalmas[j, 0] = strmas[mas[j] \* 4];

finalmas[j, 1] = strmas[mas[j] \* 4 + 1];

finalmas[j, 2] = strmas[mas[j] \* 4 + 2];

finalmas[j, 3] = strmas[mas[j] \* 4 + 3];

}

StreamWriter sw = new StreamWriter(String.Format("Cards/Topic{0}\_card{1}.Topic{0}", top, q), false, enc);

for (int i = 0; i < 3; i++)

for (int j = 0; j < 4; j++)

{

sw.WriteLine(finalmas[i, j]);

}

sw.Close();

}

private void Button\_Click\_Prov(object sender, RoutedEventArgs e)

{

string name = (string)ListBox.SelectedItem;

if(name==null)

{

MessageBox.Show("Выберите cтудента!!!", "", MessageBoxButton.OK);

return;

}

DirectoryInfo d = new DirectoryInfo("Answers/");

FileInfo[] mas = d.GetFiles(name + "\*.otvet");

int resoult = 0;

for (int i = 0; i < mas.Length; i++)

{

string res;

StreamReader r = new StreamReader(mas[i].FullName);

res = r.ReadToEnd();

r.Close();

string[] mres = res.Split(new[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);

for (int j = 0; j < 3; j++)

{

if (mres[j][mres[j].Length - 1] == '$')

resoult += 3;

}

}

string allres = String.Format(

"Студент {0} ответил на {1} билет(-ов) и набрал {2} балла(-ов) (один правильный ответ - 3 балла)", name, mas.Length, resoult);

MessageBox.Show(allres, "", MessageBoxButton.OK);

int count = 0;

StreamReader sr = new StreamReader("Marks.txt");

string str;

str = sr.ReadToEnd();

sr.Close();

string[] usersres = str.Split(new[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);

int nom = usersres.Length;

for (int i = 0; i < usersres.Length; i++)

{

if (usersres[i].Split(new[] { ' ' })[1] == name)

{

count++; nom = i;

}

}

if (count == 0)

{

StreamWriter sw = new StreamWriter("Marks.txt", true);

sw.WriteLine(allres);

sw.Close();

}

if (count != 0)

{

usersres[nom] = allres;

StreamWriter sw = new StreamWriter("Marks.txt");

for (int i = 0; i < usersres.Length; i++)

{

sw.WriteLine(usersres[i]);

}

sw.Close();

}

}

}

}

StudentForm.xaml

<Window x:Class="SummerPractice.StudentForm"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:SummerPractice"

mc:Ignorable="d"

Title="Режим студента" Height="600" Width="600" WindowStartupLocation="CenterScreen">

<Window.Background>

<ImageBrush ImageSource="Images/1.jpg"/>

</Window.Background>

<Grid>

<Menu HorizontalAlignment="Center" VerticalAlignment="Center" FontFamily="Copperplate Gothic Bold" FontSize= "30" Background="Transparent" Foreground="White">

<MenuItem Header="Выбор темы" Background="Transparent" Foreground="White">

<MenuItem Name="Topic1" IsCheckable="True" IsChecked="True" Header="Тема 1" Click="Topic\_Click" ></MenuItem>

<MenuItem Name="Topic2" IsCheckable="True" Header="Тема 2" Click="Topic\_Click"></MenuItem>

<MenuItem Name="Topic3" IsCheckable="True" Header="Тема 3" Click="Topic\_Click"></MenuItem>

<Separator></Separator>

<MenuItem Name="Exit" Header="Выход" Click="Exit\_Click"></MenuItem>

</MenuItem>

<MenuItem Name="Choose" Header="Выбор билета" Click="Choose\_Click"></MenuItem>

</Menu>

</Grid>

</Window>

StudentForm.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using Microsoft.Win32;

using System.IO;

namespace SummerPractice

{

/// <summary>

/// Interaction logic for StudentForm.xaml

/// </summary>

public partial class StudentForm : Window

{

private string studentName;

public StudentForm(string name)

{

studentName = name.ToLower();

int count = 0;

InitializeComponent();

StreamReader sr = new StreamReader(@"Students.txt");

while(!sr.EndOfStream)

{

if (sr.ReadLine().ToLower() == studentName) count++;

}

sr.Close();

if (count == 0)

{

StreamWriter sw = new StreamWriter(@"Students.txt", true);

sw.WriteLine(studentName);

sw.Close();

}

}

private void Topic\_Click(object sender, RoutedEventArgs e)

{

Topic1.IsChecked = false;

Topic2.IsChecked = false;

Topic3.IsChecked = false;

((MenuItem)sender).IsChecked = true;

}

private void Exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void Choose\_Click(object sender, RoutedEventArgs e)

{

string filter = "";

if (Topic1.IsChecked)

filter = "Тема 1 (\*.Topic1)|\*.Topic1";

if (Topic2.IsChecked)

filter = "Тема 2 (\*.Topic2)|\*.Topic2";

if (Topic3.IsChecked)

filter = "Тема 3 (\*.Topic3)|\*.Topic3";

OpenFileDialog f = new OpenFileDialog();

f.InitialDirectory = "Cards";

f.Filter = filter;

if (f.ShowDialog() == true)

{

AnswerForm a = new AnswerForm(f.FileName, studentName);

a.Show();

}

}

}

}

AnswerForm.xaml

<Window x:Class="SummerPractice.AnswerForm"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:SummerPractice"

mc:Ignorable="d"

Title="Тест" SizeToContent="WidthAndHeight" WindowStartupLocation="CenterOwner" >

<StackPanel>

<Expander Header="Вопрос №1" BorderBrush="Aqua" BorderThickness="2" Padding="5" Margin="5" FontFamily="Copperplate Gothic Light" FontSize= "14">

<StackPanel>

<Label Name="question1">вопрос 1</Label>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Name="answer1">

<RadioButton HorizontalAlignment="Left" Margin="5" Name="q1a1">ответ 1</RadioButton>

<RadioButton HorizontalAlignment="Center" Margin="5" Name="q1a2">ответ 2</RadioButton>

<RadioButton HorizontalAlignment="Right" Margin="5" Name="q1a3">ответ 3</RadioButton>

</StackPanel>

</StackPanel>

</Expander>

<Expander Header="Вопрос №2" BorderBrush="Aqua" BorderThickness="2" Padding="5" Margin="5" FontFamily="Copperplate Gothic Light" FontSize= "14">

<StackPanel>

<Label Name="question2">вопрос 2</Label>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Name="answer2">

<RadioButton HorizontalAlignment="Left" Margin="5" Name="q2a1">ответ 1</RadioButton>

<RadioButton HorizontalAlignment="Center" Margin="5" Name="q2a2">ответ 2</RadioButton>

<RadioButton HorizontalAlignment="Right" Margin="5" Name="q2a3">ответ 3</RadioButton>

</StackPanel>

</StackPanel>

</Expander>

<Expander Header="Вопрос №3" BorderBrush="Aqua" BorderThickness="2" Padding="5" Margin="5" FontFamily="Copperplate Gothic Light" FontSize= "14">

<StackPanel>

<Label Name="question3">вопрос 3</Label>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Name="answer3">

<RadioButton HorizontalAlignment="Left" Margin="5" Name="q3a1">ответ 1</RadioButton>

<RadioButton HorizontalAlignment="Center" Margin="5" Name="q3a2">ответ 2</RadioButton>

<RadioButton HorizontalAlignment="Right" Margin="5" Name="q3a3">ответ 3</RadioButton>

</StackPanel>

</StackPanel>

</Expander>

<Button Margin="3" Click="Button\_Click" FontFamily="Copperplate Gothic Light" FontSize= "14">Сохранить</Button>

</StackPanel>

</Window>

AnswerForm.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.IO;

namespace SummerPractice

{

/// <summary>

/// Interaction logic for AnswerForm.xaml

/// </summary>

public partial class AnswerForm : Window

{

private string path;

private string name;

private string[] mas;

public AnswerForm(string path, string name)

{

this.name = name;

this.path = path;

InitializeComponent();

SetAll(path);

}

private void SetAll(string path)

{

string t = path.Remove(path.Length - 7, 7);

Title = t.Substring(t.Length - 12, 12);

string s;

Encoding enc = Encoding.GetEncoding(1251);

StreamReader sr = new StreamReader(path, enc);

s = sr.ReadToEnd();

mas = s.Split(new string[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);

sr.Close();

question1.Content = mas[0];

if (mas[1][mas[1].Length - 1] == '$')

q1a1.Content = mas[1].TrimEnd('$');

else

q1a1.Content = mas[1];

if (mas[2][mas[2].Length - 1] == '$')

q1a2.Content = mas[2].TrimEnd('$');

else

q1a2.Content = mas[2];

if (mas[3][mas[3].Length - 1] == '$')

q1a3.Content = mas[3].TrimEnd('$');

else

q1a3.Content = mas[3];

question2.Content = mas[4];

if (mas[5][mas[5].Length - 1] == '$')

q2a1.Content = mas[5].TrimEnd('$');

else

q2a1.Content = mas[5];

if (mas[6][mas[6].Length - 1] == '$')

q2a2.Content = mas[6].TrimEnd('$');

else

q2a2.Content = mas[6];

if (mas[7][mas[7].Length - 1] == '$')

q2a3.Content = mas[7].TrimEnd('$');

else

q2a3.Content = mas[7];

question3.Content = mas[8];

if (mas[9][mas[9].Length - 1] == '$')

q3a1.Content = mas[9].TrimEnd('$');

else

q3a1.Content = mas[9];

if (mas[10][mas[10].Length - 1] == '$')

q3a2.Content = mas[10].TrimEnd('$');

else

q3a2.Content = mas[10];

if (mas[11][mas[11].Length - 1] == '$')

q3a3.Content = mas[11].TrimEnd('$');

else

q3a3.Content = mas[11];

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

string f = (new FileInfo(path)).Name;

StreamWriter wr = new StreamWriter("Answers/" + name + "\_" + f.Substring(0, 12) + ".otvet", false);

if ((bool)q1a1.IsChecked)

{

if (mas[1][mas[1].Length - 1] == '$')

{

wr.WriteLine("1$");

}

else

wr.WriteLine("1");

}

if ((bool)q1a2.IsChecked)

{

if (mas[2][mas[2].Length - 1] == '$')

{

wr.WriteLine("2$");

}

else

wr.WriteLine("2");

}

if ((bool)q1a3.IsChecked)

{

if (mas[3][mas[3].Length - 1] == '$')

{

wr.WriteLine("3$");

}

else

wr.WriteLine("3");

}

if ((bool)q2a1.IsChecked)

{

if (mas[5][mas[5].Length - 1] == '$')

{

wr.WriteLine("1$");

}

else

wr.WriteLine("1");

}

if ((bool)q2a2.IsChecked)

{

if (mas[6][mas[6].Length - 1] == '$')

{

wr.WriteLine("2$");

}

else

wr.WriteLine("2");

}

if ((bool)q2a3.IsChecked)

{

if (mas[7][mas[7].Length - 1] == '$')

{

wr.WriteLine("3$");

}

else

wr.WriteLine("3");

}

if ((bool)q3a1.IsChecked)

{

if (mas[9][mas[9].Length - 1] == '$')

{

wr.WriteLine("1$");

}

else

wr.WriteLine("1");

}

if ((bool)q3a2.IsChecked)

{

if (mas[10][mas[10].Length - 1] == '$')

{

wr.WriteLine("2$");

}

else

wr.WriteLine("2");

}

if ((bool)q3a3.IsChecked)

{

if (mas[11][mas[11].Length - 1] == '$')

{

wr.WriteLine("3$");

}

else

wr.WriteLine("3");

}

wr.Close();

Close();

}

}

}