**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное образовательное бюджетное учреждение высшего образования**

**«Вологодский государственный университет»**

**(ВоГУ)**

* 1. **ОБЪЕКТНО-ОРЕНТИРОВАННОЕ программирование**
  2. **лабораторнАЯ РАБОТА №7**

**«Язык программирования C#. Разработка простых графических и сетевых программ»**

**Направление подготовки: 09.03.01 Информатика и вычислительная техника**

**Направленность (профиль): Программное обеспечение средств вычислительной техники и автоматизированных систем**

**Форма обучения: очная**

**Институт: Математики, естественных и компьютерных наук**

**Кафедра: Автоматики и вычислительной техники**

**Группа: 4Б09 ПО-41**

**Студент: Махонин М.Н.**

**Руководитель: Сорокин А.Н.**

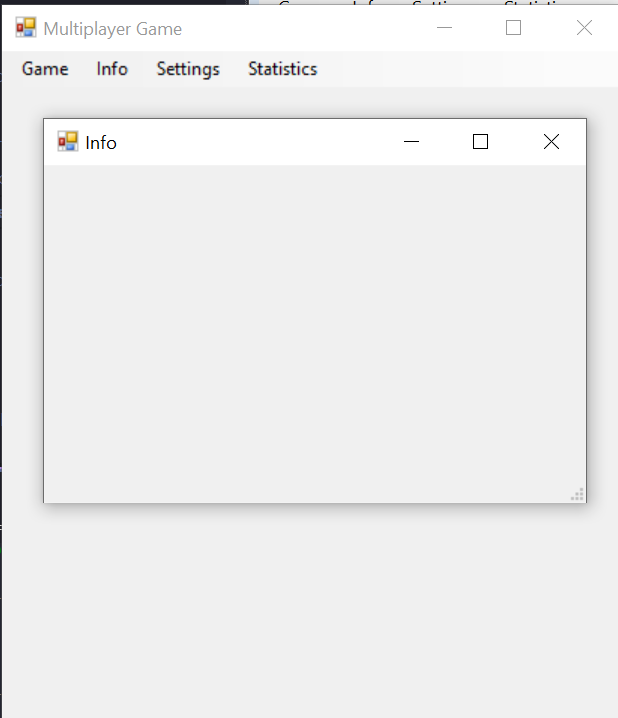
Вологда

2021 г.

**Цель работы:** научиться создавать простые графические и сетевые приложения на языке программирования C#.

1. Добавить пункт меню для вызова диалогового окна с информацией о программе.

Форма диалогового окна с информацией.



Код окна информации

public partial class Info : Form

{

public Info()

{

InitializeComponent();

}

}

Вызывается данная форма при помощи кнопки в меню Info.

Код данной кнопки.

private void infoToolStripMenuItem\_Click(Object sender, EventArgs e)

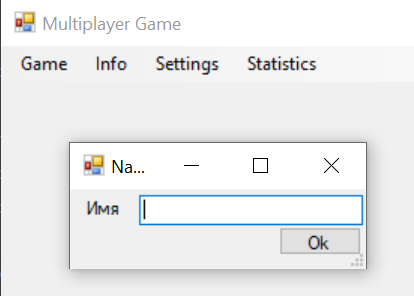
{

Info dlg = new Info();

dlg.ShowDialog();

}

1. Добавить в программу возможность хранения имени игрока. При создании нового игрока пользователь должен указать имя игрока.



При создании сервера или подключения к другому серверу предлагается ввести имя игрока.

public partial class NameForm : Form

{

public string InputText

{

get => textBoxName.Text;

set => textBoxName.Text = value;

}

public NameForm() => InitializeComponent();

private void buttonOk\_Click(Object sender, EventArgs e)

{

InputText = textBoxName.Text;

Close();

}

}

Вызов формы(Player.cs).

public Player(Game g)

{

SetName();

game = g;

Random rnd = new Random();

x = 20 + rnd.Next(100);

y = 20 + rnd.Next(100);

}

private void SetName()

{

NameForm dlg = new NameForm();

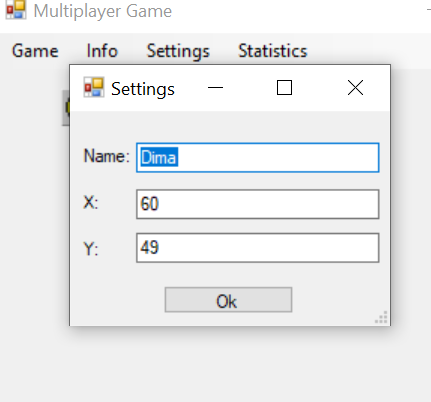
dlg.ShowDialog();

Name = dlg.InputText;

}

1. Добавить пункт меню для вызова диалогового окна с настройками игрока. В диалоговом окне пользователь может поменять имя игрока и его координаты.

Форма данного меню.



При вызове данной формы в нее передается параметр в виде класса Player, для вывода его полей на экран.

public partial class SettingsForm : Form

{

public string Name

{

get => textBoxName.Text;

set => textBoxName.Text = value;

}

public string X

{

get => textBoxX.Text;

set => textBoxX.Text = value;

}

public string Y

{

get => textBoxY.Text;

set => textBoxY.Text = value;

}

public SettingsForm()

{

InitializeComponent();

}

public void GetSettings(Player player)

{

Name = player.Name;

X = player.X.ToString();

Y = player.Y.ToString();

}

private void buttonOk\_Click(Object sender, EventArgs e)

{

Name = textBoxName.Text;

X = textBoxX.Text;

Y = textBoxY.Text;

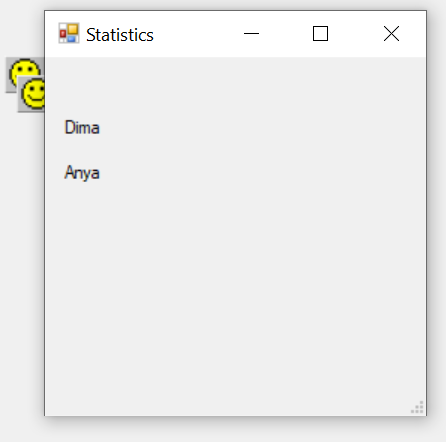
Close();

}

}

Вызов данной формы происходит через копку Settings в меню.

1. Добавить пункт меню для вывода статистики игры. В самом простом варианте можно вывести список всех игроков.



В данном меню отображаются все игроки на сервере.

При вызове данной формы в нее передается в качестве параметра ArrayList объектов Player.

public partial class StatisticsForm : Form

{

private int lovationX = 10;

private int lovationY = 10;

public StatisticsForm(ArrayList players)

{

InitializeComponent();

LoadStatus(players);

}

private void LoadStatus(ArrayList players)

{

foreach(var player in players)

{

lovationY += 30;

Label label = new Label();

var point = new Point(lovationX, lovationY);

label.Size = new Size(150, 20);

label.Location = point;

label.Text = ((Player)player).Name;

this.Controls.Add(label);

}

}

}

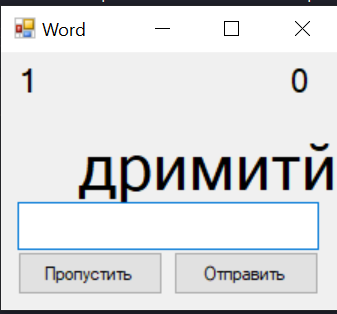
Вызов происходит через кнопку Statistics в меню.

**Дополнительное задание**

**Вариант 2**

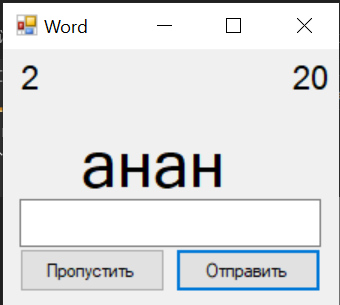
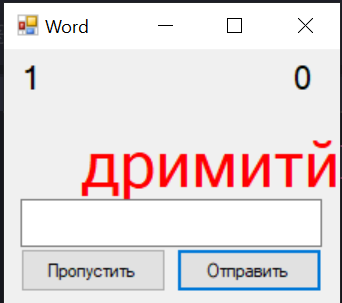
Создайте диалоговую игру «Угадай слово», позволяющую пользователю угадывать загаданное компьютером слово. Слова хранятся в файле или базе данных.

Ниже представлено основное окно приложения

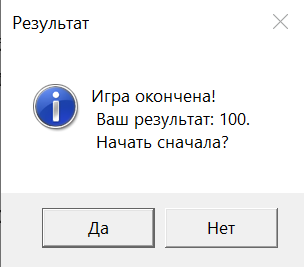


Слова считываются из файла "phrases.txt". В левом верхнем углу показывается слово по счету, в правом количество очков. Необходимо ввести загаданное слово и нажать кнопку «Отправить». Если не удается угадать слово, то кнопку «Пропустить» (этом случае количество очков не увеличивается, показывается следующее слово).

Если пользователь угадал слово, то количество очков увеличивается на 20, показывается следующее слово, если не угадал, то слово подсвечивается красным.

После того, как все слова будут выведены, пользователю выходит сообщение об окончании игры и предлагается либо выйти, либо начать заново.



Код программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace oopLab7Word

{

public partial class Form1 : Form

{

private List<string> words = new List<string>();

private int count = 0;

private int point = 0;

private int numWord = 0;

private Random rndWord = new Random();

public Form1()

{

InitializeComponent();

LoadWords();

UpdateWord();

}

private void LoadWords()

{

if(File.Exists("phrases.txt"))

{

string[] readText = File.ReadAllLines("phrases.txt", Encoding.UTF8);

words.AddRange(readText);

}

else

{

words.Add("прыгать");

words.Add("гидро");

words.Add("котенок");

words.Add("лист");

words.Add("жизнь");

}

}

private void UpdateWord()

{

if(count == 5)

{

DialogResult result = MessageBox.Show(

$"Игра окончена!\n Ваш результат: {point}. \n Начать сначала?",

"Результат",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if(result == DialogResult.Yes)

{

count = 0;

point = 0;

LoadWords();

UpdateWord();

}

else Close();

return;

}

labelCount.Text = (count + 1).ToString();

var rnd = new Random();

numWord = rndWord.Next(0, words.Count - 1);

var shuffle = string.Join("", words[numWord].OrderBy(x => rnd.Next()));

labelWord.Text = shuffle;

}

private void buttonSend\_Click(Object sender, EventArgs e)

{

if(words[numWord].ToLower() == textBoxWord.Text.ToLower())

{

labelWord.ForeColor = Color.Black;

count++;

point += 20;

words.Remove(words[numWord]);

UpdateWord();

labelPoint.Text = point.ToString();

textBoxWord.Text = "";

}

else

{

labelWord.ForeColor = Color.Red;

textBoxWord.Text = "";

}

}

private void buttonSkip\_Click(Object sender, EventArgs e)

{

words.Remove(words[numWord]);

count++;

UpdateWord();

labelWord.ForeColor = Color.Black;

}

}

}

**Контрольные вопросы**

1. **Что понимается под терминами «делегат» и «событие»? Для чего применяются делегаты и события?**

*Делегатом* является объект ссылочного типа, предоставляющий ссылки на метод или группу методов приложения.

* Делегаты имеют следующие свойства:
* Делегаты похожи на указатели функций в C++, но являются объектно-ориентированными и типобезопасными
* Делегаты допускают передачу методов в качестве параметров.
* Делегаты можно использовать для определения методов обратного вызова (callback'ов).
* Делегаты можно связывать друг с другом; например, при появлении одного события можно вызывать несколько методов.

Делегату можно назначить любой метод из любого доступного класса или структуры, соответствующей типу делегата. Этот метод должен быть статическим методом или методом экземпляра. Это позволяет программно изменять вызовы метода, а также включать новый код в существующие классы.

*Событие* представляет собой автоматическое уведомление о том, что произошло некоторое действие. События действуют по принципу работы паттерна проектирования "издатель-подписчик": объект, проявляющий интерес к событию, регистрирует обработчик этого события. Когда же событие происходит, вызываются все зарегистрированные обработчики этого события. Обработчики событий обычно представлены делегатами.

События похожи на свойства: внутри они содержат поле делегата, доступ напрямую к которому запрещен. Публичное поле делегата (или публичное свойство) может привести к тому, что список обработчиков события может быть очищен другим объектом, или что событие будет вызвано извне — в то время как мы хотим вызывать его только из исходного объекта.

Свойства событий:

* Издатель определяет, когда возникает событие; подписчики определяют, какое действие выполняется в ответ на событие.
* У события может быть несколько подписчиков. Подписчик может обрабатывать несколько событий от нескольких издателей.
* События, не имеющие подписчиков, никогда не возникают.
* Обычно события используются для оповещения о действиях пользователя, например нажатиях кнопок или выборе пунктов меню в графических пользовательских интерфейсах.
* Если событие имеет несколько подписчиков, при возникновении события обработчики событий вызываются синхронно.
* В библиотеке классов .NET Framework события основываются на делегате EventHandler и базовом классе EventArgs.

1. **Что понимается под термином «пространство имен» в языке C#? Приведите примеры описания и использования пространств имен.**

Обычно определяемые классы и другие типы в .NET не существуют сами по себе, а заключаются в специальные контейнеры - пространства имен. Пространства имен позволяют организовать код программы в логические блоки, позволяют объединить и отделить от остального кода некоторую функциональность, которая связана некоторой общей идеей или которая выполняет определенную задачу.

В пример можно привести мое дополнительное задание, там все классы объединены в пространство имен приложения, так же мы подключаем другие пространства имен используя using.