Министерство образования Российской Федерации

Отчёт по

**Контрольная работа**

**Разработка игрового приложения**

Выполнил:

Проверил:

2022г.

**Содержание**

Цель работы. Задание 3

Разработка приложения 4

Основная логика работы программы 5

Разработанные элементы (классы, перечисления). 6

Разработанные окна. 10

Примеры работы программы. 12

Заключение 15

Список литературы. 16

**Контрольная работа**

**Разработка игрового приложения**

**Цель:**

Целью данной работы является закрепление навыков, полученных в ходе выполнения предыдущих лабораторных работ.

**Задание:**

Разработать и реализовать приложение, состоящее из двух модулей:

1. «Генератор уровней» – набор функций для генерации поля заданного размера и проверки допустимости действий пользователя

* Размеры поля являются переменными параметрами и задаются пользователем.
* Логически поле представлено в виде двумерного массива.
* Каждое действие пользователя приводит к изменению состояния поля.
* После каждого действия, совершаемого над полем, должны проверяться условия победы и поражения.

1. «Графический интерфейс» – окно WPF и связанные с ней функции для отображения игрового поля и обработки действий пользователя.

**Вариант 3**:

Разработать и реализовать игру «Память».

Правила игры «Память» <http://testbrain.ru/tests/viewMemory>

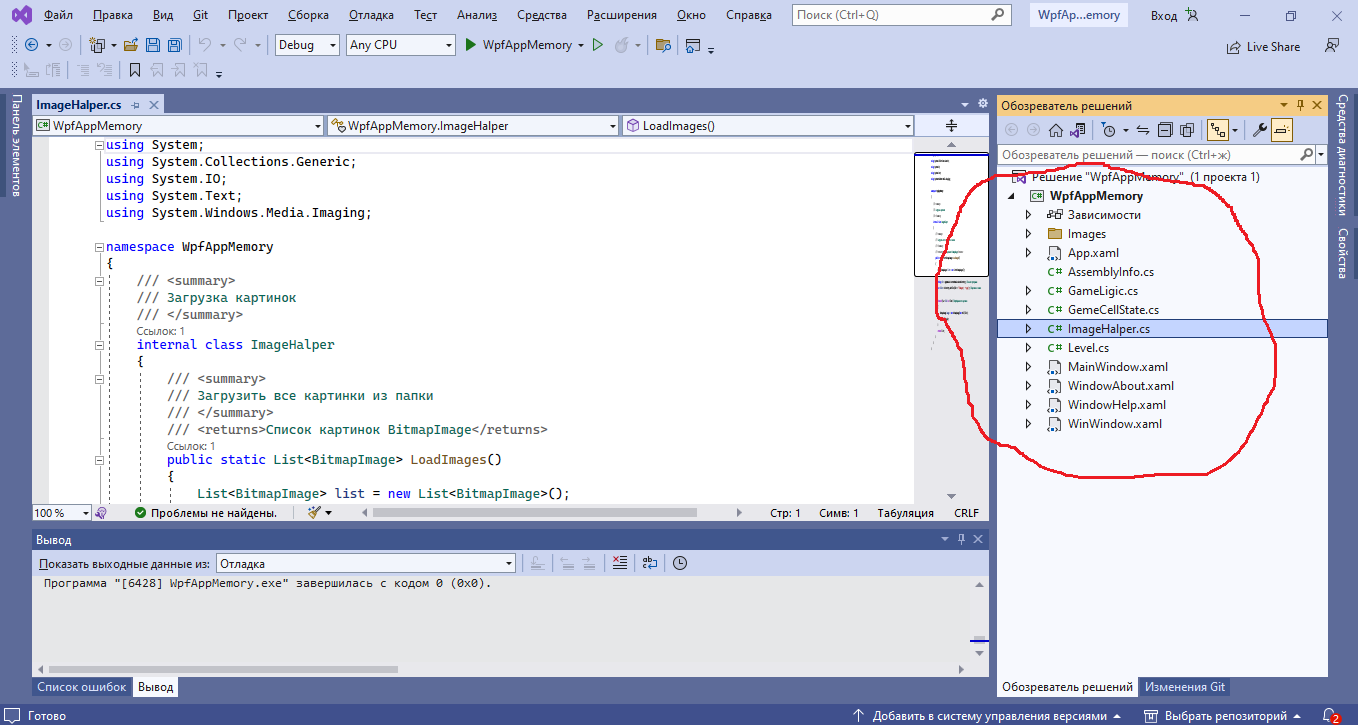
**Разработка приложения**

Целью данной работы является разработка WPF-приложения, которое представляет из себя игру «Память». В рамках реализации WPF-приложения была спроектирована и разработана основная логическая часть приложения (классы), а также визуальная часть (Окна, написанные с помощью XAML).

Также одним из требований, предъявляемых к разрабатываемой игре, является создание удобного, не перегруженного лишней информацией и функционалом пользовательского интерфейса, обеспечивающего легкое восприятие и интуитивно понятное назначение элементов.

Программа предназначена для запуска в OC Windows, среда разработки Microsoft Visual Studio, язык программирования C#.

Структура проекта показана на рисунке



**Основная логика работы программы.**

Картинки, используемые для игрового поля, хранятся в папке images.

В программе используется два основных массива:

1) Первый хранит номера картинок для каждой ячейки. Заполняется матрица случайным образом, при этом учитывается, что картинки на игровое поле должны обязательно помещаться парами.

2) Второй массив хранит состояние ячейки. Каждая ячейка может находится в одном из трёх состояний: закрытая, открытая, приоткрытая на текущий ход.

Алгоритм проверки окончания игры:

1. Перебираем все ячейки игрового поля

2. Поверяем состояние каждой ячейки

2.1. Если все ячейки открыты, значит игра окончена

2.2. Иначе (т.е. если есть хотя бы одна закрытая ячейка) игра ещё не окончена

В программе ограничимся пятью размерами доски. Делать доску больше размера 12х12 нежелательно (это максимальный размер доски в программе), т.к. тогда такая доска не умещается на экран 19” монитора.

Кроме того, в программе есть счётчик сделанных ходов и вычисляется время игры.

**Разработанные элементы.**

В ходе выполнения работы были разработаны следующие элементы (их список виден на картинке со структурой проекта):

1. Перечисление Level, которое представляет уровни игры и размеры поля для каждого уровня

/// <summary>

/// Уровень игры

/// </summary>

public enum Level

{

first=4,

nextFirst = 6,

second =8,

nextSecond = 10,

third =12

}

2. Перечисление GemeCellState, которое используется для того, чтобы указать в каком состоянии находится ячейка игрового поля

/// <summary>

/// Состояние поля с картинкой.

/// </summary>

public enum GemeCellState

{

/// <summary>

/// закрыта

/// </summary>

closed,

/// <summary>

/// открыта

/// </summary>

opened,

/// <summary>

/// приоткрыта на один шаг

/// </summary>

visible

}

3. Вспомогательный класс ImageHelper, который содержит только один статический метод LoadImages, в задачу которого входит загрузка всех картинок из папки «Images». Все эти картинки используются в качестве картинок для ячеек игрового поля.

/// <summary>

/// Загрузка картинок

/// </summary>

internal class ImageHalper

{

/// <summary>

/// Загрузить все картинки из папки

/// </summary>

/// <returns>Список картинок BitmapImage</returns>

public static List<BitmapImage> LoadImages()

{

List<BitmapImage> list = new List<BitmapImage>();

string dir = AppDomain.CurrentDomain.BaseDirectory; //каталог программы

var files = Directory.GetFiles(dir + "\\Images", "\*.png"); //картинки из папки

foreach (var file in files) //перебираем все картинки

{

BitmapImage image = new BitmapImage(new Uri(file));

list.Add(image);

}

return list;

}

}

4. Класс GameLogic, в который помещена вся логика игры.

Игровое поле представлено двумя массивами

1) Массив gameBord, который содержит номер картинки

/// <summary>

/// Матрица игрового поля

/// </summary>

private int[,] gameBord;

2) Массив gameBordCellState, который содержит информацию, в каком из состояний находится ячейка

/// <summary>

/// Матрица состояний ячеек игрового поля.

/// (Указывает в каком из трёх состояний находится каждая ячейка)

/// </summary>

GemeCellState[,] gameBordCellsState;

Новая игра запускается вызовом метода StartGame, в котором очищается старая доска, создаётся новая доска и запускается таймер игры.

Игру можно остановить вызовом метода StopGame, в котором останавливается таймер и выводится соответствующее сообщение.

Проверка того, открыты ли все ячейки проходит в методе CheckEndGame(). Если есть хотя бы одна закрытая ячейка, игра продолжается, иначе (все ячейки открыты) игра останавливается.

Логика обработки нажатия мышкой пользователем на конкретную ячейку помещена в метод KeyPrerssed, в который в качестве параметров передаётся координата нажатой ячейки.

Поскольку код программы снабжён подробными комментариями, то нет необходимости расписывать каждый метод подробно.

5. Классы окон WPF.

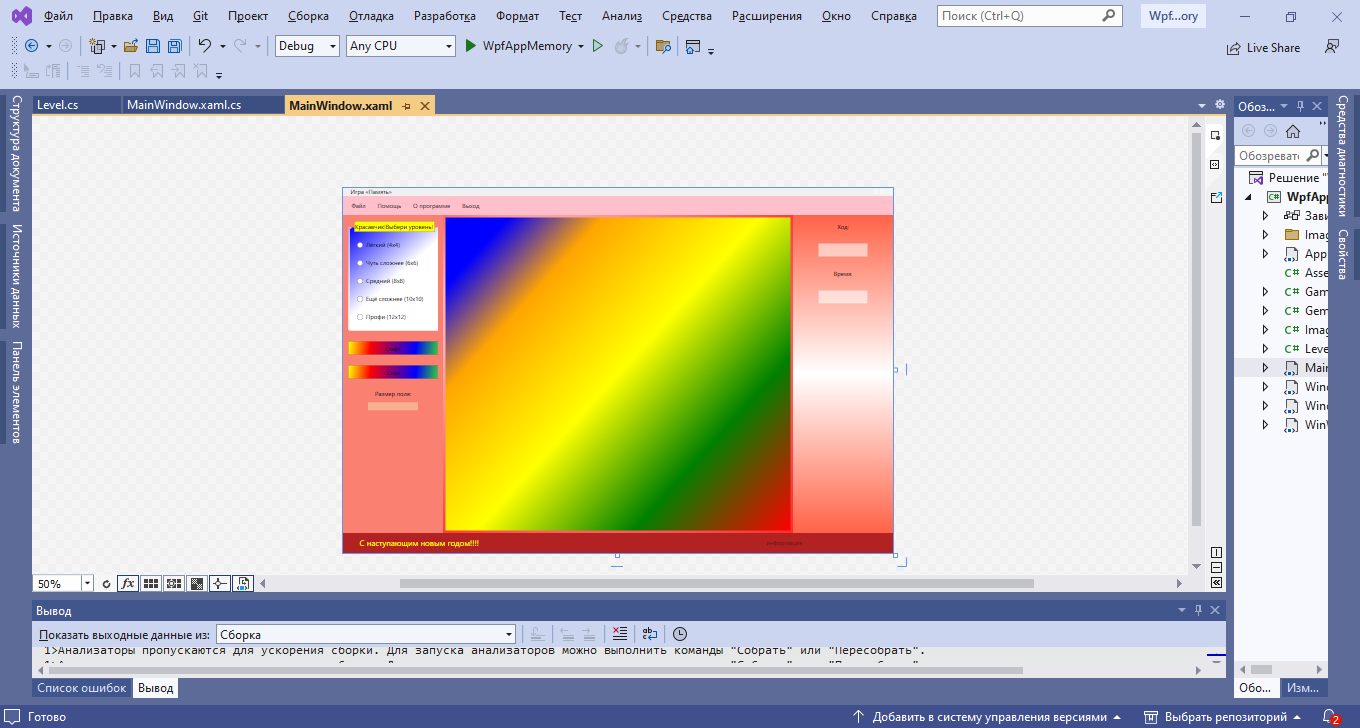
Каждое из окон программы кроме разметки XAML имеют и свои собственные классы. Методы, реализованные в этих классах отвечают за отрисовку доски, вывод информации для пользователя (время игры, сообщения) и обработку действий пользователя (нажатие клавиш, выбор пунктов меню, нажатие на ячейках игрового поля и т. д.).

Поскольку код программы снабжён подробными комментариями и методы сделаны небольшими, то нет необходимости расписывать каждый метод подробно.

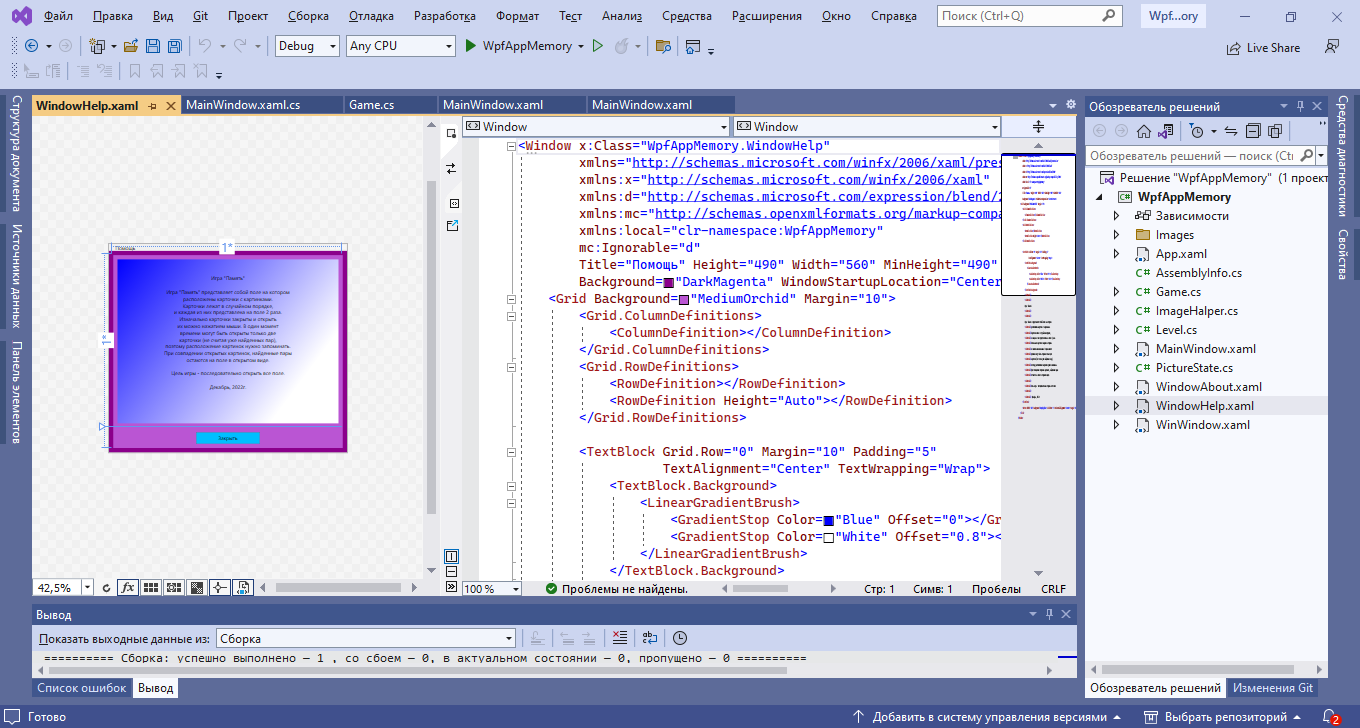
**Разработанные окна.**

Были разработаны следующие окна.

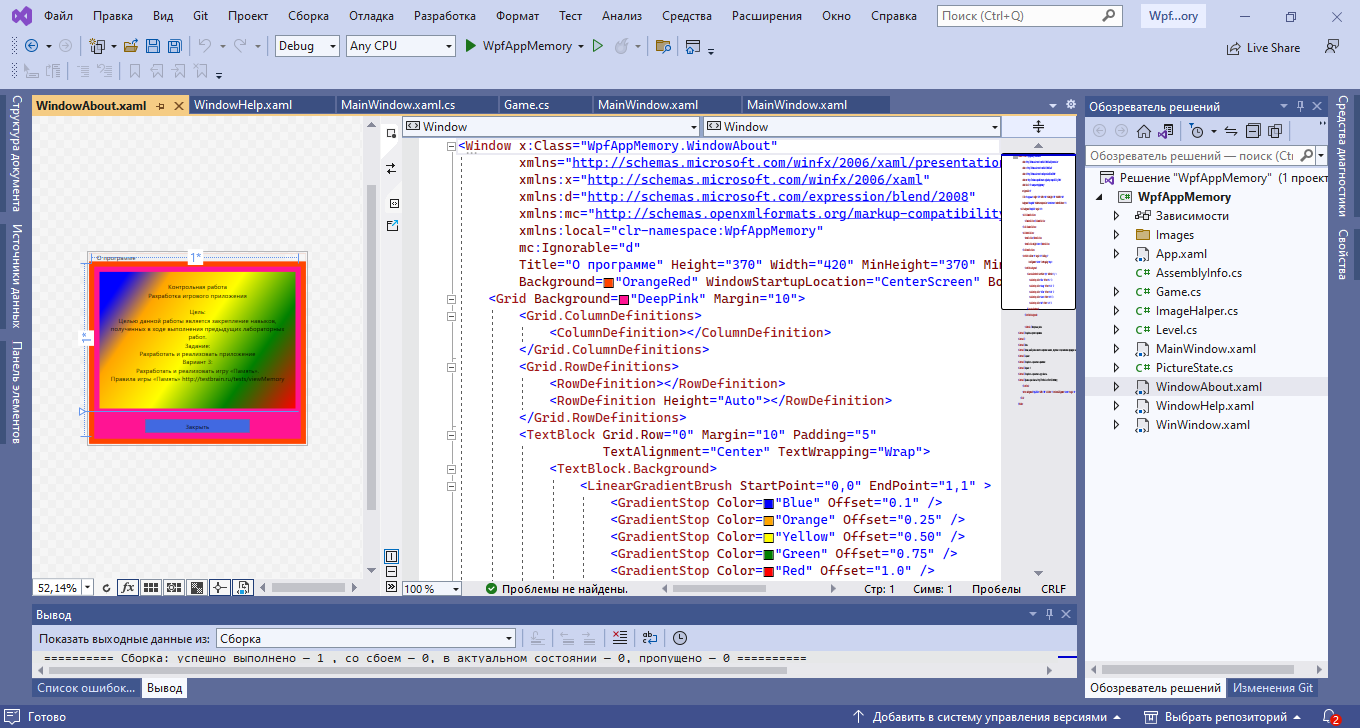
1.Главное окно приложения



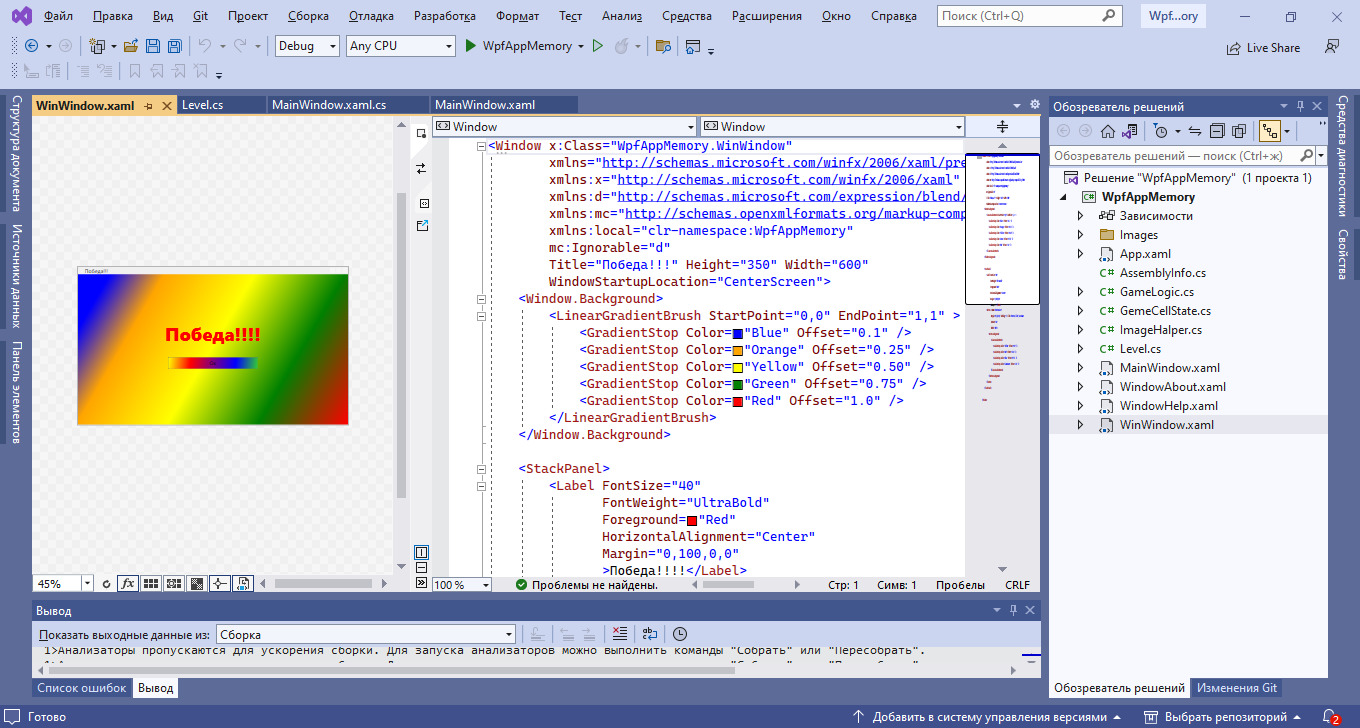
2.Окно помощи



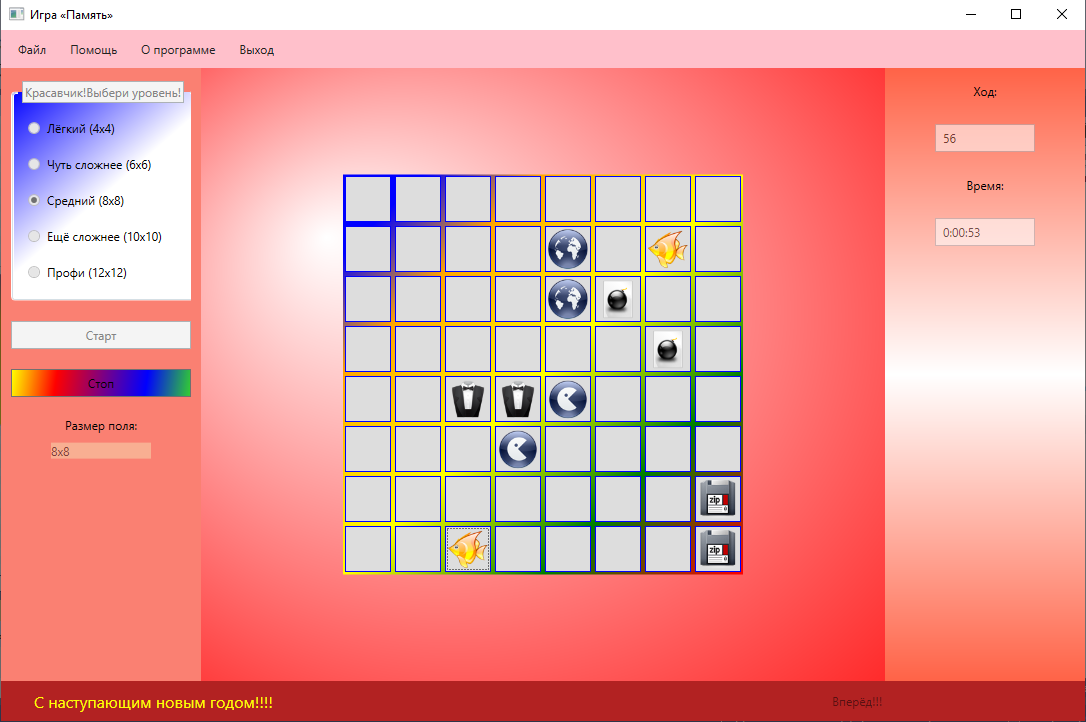
3.Окно о программе

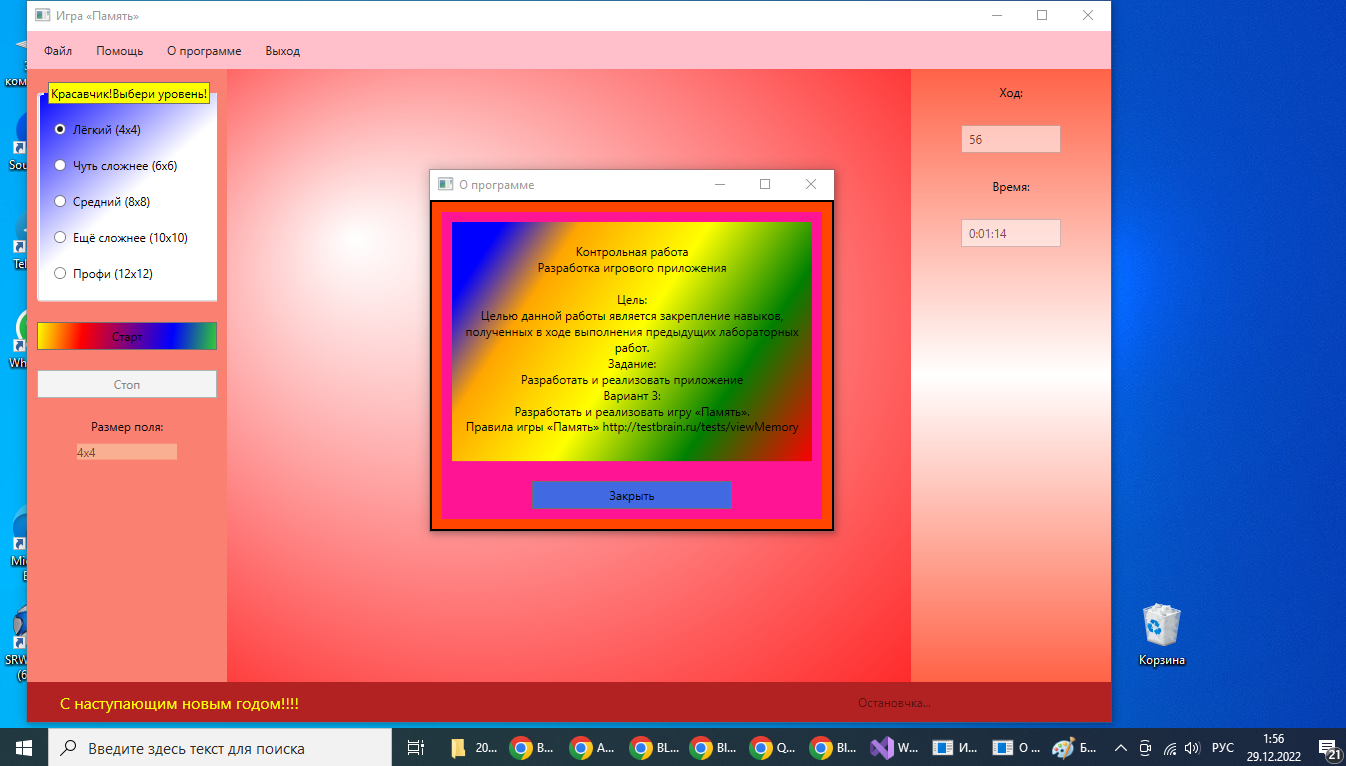


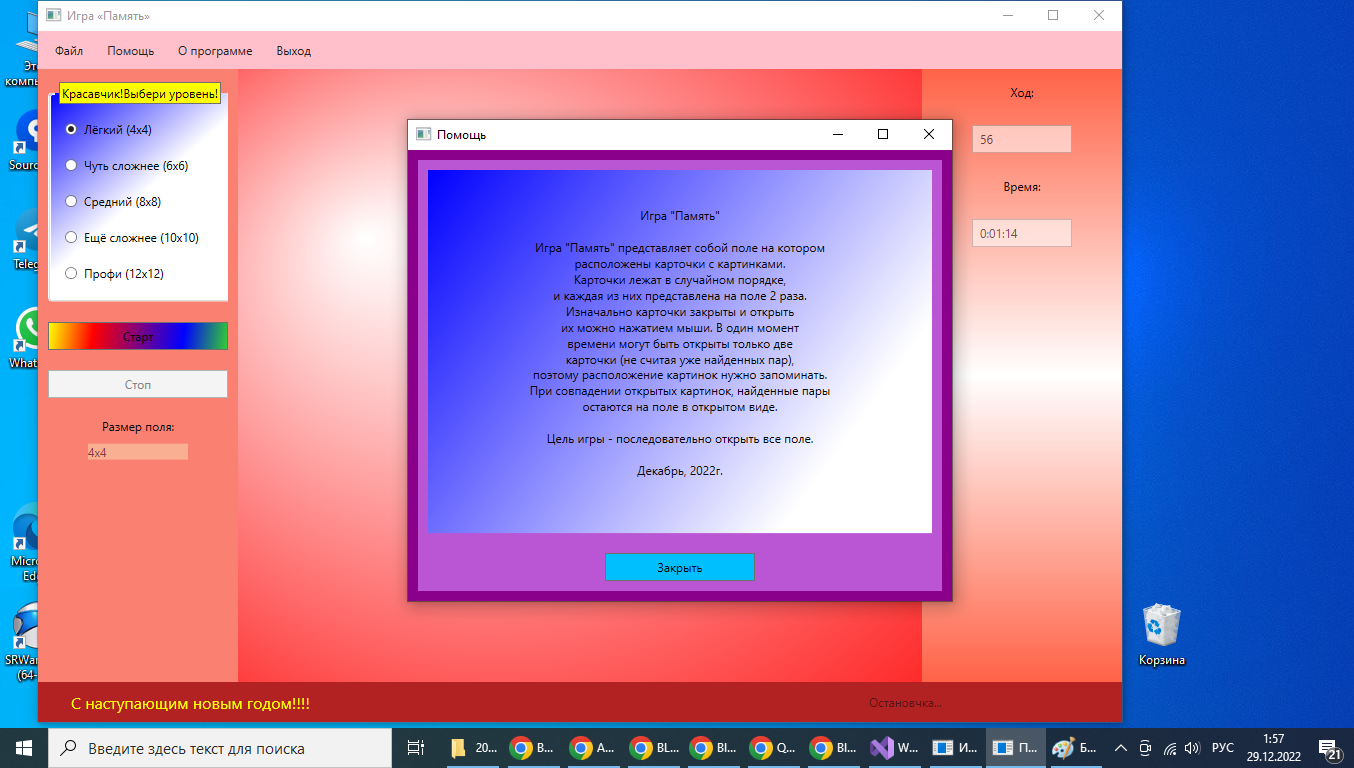
4.Окно победы



**Примеры работы программы.**







**Заключение**

В ходе выполнения работы были пройдены следующие этапы:

1. Изучены основы языка C#
2. Изучен язык разметки XAML
3. Проанализирована поставленная задача
4. Были разработаны классы, необходимые для реализации идеи игры
5. Разработаны окна программы
6. Проведены отладка и тестирование. Тестирование программы прошло успешно

Так же при написании работы была изучена соответствующая литература по разработке приложений на языке программирования C# и был получен практический опыт разработки WPF-приложения на языке C#.

По результатам работы составлен отчёт.

**Требования к техническому и программному обеспечению**

Требования к программному обеспечению:

− Операционная система Windows 10 и выше;

− Net 6.0 и выше;

Требования к аппаратному обеспечения:

− Процессор не ниже Intel Core i3 2.5 GHz;

− Свободного места на жёстком диск не менее 100Мб;

− ОЗУ не менее 4 Гб.

**Список литературы**

1. А. Васильев Программирование на С# для начинающих. Основные сведения. – изд-во «Бомбора», 2018. – 586 стр.
2. Вагнер, Б. Эффективное программирование на C#. 50 способов улучшения кода. / Б. Вагнер. -М.: Вильямс И.Д., 2017. -224 c.8.
3. Джуст Виссер Разработка обслуживаемых программ на языке C#. – изд-во ДМК-Пресс, 2017. – 192 стр.
4. Кормен Т. Ривест Р. Лейзерсон Ч. Алгоритмы. Построение и анализ – пер. Красиков И. В. изд-во Вильямс, 2016. – 1328 стр.
5. Краткий обзор языка C# — Официальный сайт Microsoft — 2019. — С. 1 — [Электронный ресурс] – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/index (дата обращения: 12.04.2022).
6. Троелсен, Э. Язык программирования С# 5.0 и платформа .NET 4.5 / Э. Троелсен; Пер. с англ. Ю.Н. Артеменко. —М.: Вильямс, 2016. —1312 c.