Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет математики и информационных технологий

Кафедра компьютерной безопасности и математического обеспечения

информационных систем

**КУРСОВАЯ РАБОТА**

по дисциплине «Языки программирования»

**Разработка программы-симулятора «Игра в 15»**

ОГУ 10.05.01.3016.017 О

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Н. Полежаев

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016г.

Исполнитель

студент гр. 15КБ(с)РЗПО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Фадеева

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016г.

Оренбург 2016

**Содержание**

[Введение 3](#_Toc470512182)

[1 Техническое задание 5](#_Toc470512183)

[2 Теоретические сведения 6](#_Toc470512184)

[2.1 Основные сведения 6](#_Toc470512185)

[2.2 Математическое описание 6](#_Toc470512186)

[3 Описание алгоритмов 8](#_Toc470512187)

[3.1 Файлы проекта и их основное назначение 8](#_Toc470512188)

[3.2 Описание модулей, в которых реализована программа 8](#_Toc470512189)

[3.2.1 Модуль MainWindow.xaml.cs 8](#_Toc470512190)

[4 Руководство пользователя 11](#_Toc470512191)

[4.1 Сведения о файлах программы и способе их установки 11](#_Toc470512192)

[4.2 Интерфейс программы 11](#_Toc470512193)

[Заключение 14](#_Toc470512194)

[Список использованных источников 15](#_Toc470512195)

[Приложение А 16](#_Toc470512196)

# **Введение**

Развитие вычислительной техники тесно связано с развитием игровой индустрии.

50-ые годы двадцатого века являются отправной точкой в разработке компьютерных игр. На сегодняшний день невозможно прийти к выводу, кто же был их создателем. Считают, что первую в мире компьютерную видеоигру создал сотрудник лаборатории Брукхвена, физик-ядерщик - Вилли Хигинботэм (Willy Higinbotham).

Для демонстрации возможностей своей научной лаборатории он решил придумать развлечение, связанное с его интересами – с теннисом. Используя экран осциллоскопа, за 3 недели он создал видеоигру «Теннис для двоих» (Tennis for Two). Игра была чрезвычайно популярна, но существовала до тех пор, пока осциллоскоп ни понадобился для работы.

Хигинботэм посчитал, что не сделал выдающегося открытия, поэтому не запатентовал устройство, что стало ошибкой, как показало время.

Однако «отцом» видеоигр называют британского ученого Артура Дугласа (A.S. Douglas). В 1952 году он защищал диссертацию, приложением к которой была разработка игровой компьютерной программы крестики-нолики. По сути, она служила лишь для представления возможностей компьютера, и не была игрой в общем понимании этого слова.

История компьютерных игр развивалась достаточно медленно.

В 80-ые годы персональные компьютеры стали более доступными по цене, что способствовало развитию рынка компьютерных игр. Считается, что именно в 1980-х наступила эпоха видеоигр. В этом время появилось множество компаний, занимающихся их разработкой.

Начиная с 90-х, данная индустрия ускоренно набирала обороты. В это время появилась первая игра в жанре шутер ‒ Doom, выпущенная компанией Id Software.

В начале XXI века развитие игр не останавливается. Ежегодно выпускаются тысячи игр в огромных масштабах.

Развитие игр способствует развитию компьютеров, так как запросы пользователей к играм всё выше. Для улучшения геймплея, повышения детализованности и реалистичности графики требуются более мощные процессоры, видеокарты, требуется больший объем оперативной памяти. К тому же игровая индустрия способствует развитию рынка труда – различные компании и разработчики ищут хороших программистов для описания кода игры, сценаристов для создания сюжета, PR-менеджеров для рекламы и т.д.

Игровая индустрия имеет свои особенности, и она не прекращает развиваться и расширяться.

Разработанная программа имеет непосредственное отношение к игровой индустрии, чем объясняется актуальность данной курсовой работы.

***Целью*** *данной курсовой работы является создание логической игры, являющейся прекрасным тренажёром для развития логического мышления, усидчивости, логики и способности к анализу.*

# **1 Техническое задание**

Разработать программу, реализующую данную игру на персональном компьютере по следующим правилам:

- Игровое поле размером 4х4 размечено на клетки.

- Каждая клетка имеет уникальное числовое значение от 1 до 15, одна клетка остается пустая.

- Клетка может перемещаться на место пустой клетки, если она находится непосредственно рядом с ней.

- Игра заканчивается, если значения в клетках упорядочены по возрастанию слева направо и сверху вниз.

**Исходное задание:**

- Должен быть реализован пользовательский интерфейс.

- Программа должна осуществлять пошаговую и непрерывную анимацию игры.

- Реализовать показ времени, за которое пройдена игра.

- Реализовать алгоритм для генерирования только решаемых головоломок.

# **2 Теоретические сведения**

## **2.1 Основные сведения**

«Игра в 15» (она же «Пятнашки») – популярная и всем известная логическая игра, придуманная в 1878 году Ноем Чепмэном.

При изучении предметной области, выяснилось, что «пятнашки» на первый взгляд простая игра, но для ее реализации необходимо обратится не только к визуальной реализации игры, но и к её математическому описанию.

Суть самой игры заключается в следующем: на экране расположено поле, разбитое на 16 клеток. Клетки заполнены неповторяющимися числами от 1 до 15, в случайном порядке, оставляя одну из позиций (шестнадцатую) пустой.

В общем виде данное табло можно представить в виде таблицы 1.

Таблица 1 – Образец игрового поля

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 12 | 15 | 2 |
| 5 | 13 | 7 | 3 |
| 8 | 6 | 1 | 10 |
| 11 | 14 | 9 |  |

Игрок должен перемещать по одной клетке, соседствующей с пустой, на место свободной. Так происходит до тех пор, пока игрок не упорядочит значения по возрастанию, начиная слева направо и сверху вниз (таблица 2).

Таблица 2 – Правильное заполнение игрового поля

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 4 | 2 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 |  |

## **2.2 Математическое описание**

Математически доказано, что ровно половина из всех возможных 16! (=20 922 789 888 000) начальных расположений значений по клеткам являются нерешаемыми, то есть такие комбинации невозможно привести к собранному виду.

Пусть клетка с числом i расположена до (если считать слева направо и сверху вниз) k клеток со значениями меньшими i. Будем считать ni=k, то есть, если после костяшки с i-м числом нет чисел, меньших i, то k=0. Также введем число e ‒ номер ряда пустой клетки (считая с 1). Если сумма

(1)

является нечетной, то решения головоломки не существует.

# **3 Описание алгоритмов**

## **3.1 Файлы проекта и их основное назначение**

О составе проекта программы можно узнать из таблицы 3.

Таблица 3 – Список файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержимое и назначение |
| MainWindow.xaml.cs | Главное окно программы. |
| App.xaml.cs | Файл кода программной части для App.xaml. |
| MainWindow.xaml | Содержит разметку главного окна. |
| App.xaml | Содержит объявления ресурсов приложения. |
| Rules.txt | Содержит текст, с описанием игры и правил. |

## **3.2 Описание модулей, в которых реализована программа**

### 3.2.1 Модуль MainWindow.xaml.cs

Данный модуль содержит класс MainWindow, поля и методы которого отображены в таблицах 4 – 7. Более подробное описание содержится в текстах программы в приложении А.

Таблица 4 – Собственные поля класса MainWindow

|  |  |
| --- | --- |
| Поле | Назначение |
| private List<Button> ListOfButtons | Лист динамически создаваемых кнопок игры в главном окне. |
| private bool btnMixIsPressed | Поле, служащее флагом для определения, нажата ли кнопка «Начать игру». |
| private int myTime | Поле, в котором хранится время (в секундах), за которое пройдена игра. |
| private string strRules | Поле, в котором хранится текст с правилами игры. |
| private DispatcherTimer timer | Поле для работы с событиями таймера. |
| private int y | Переменная для установления значения по умолчанию кнопки игрового поля. |
| private FileStream fsRules | Переменная для чтения из файла. |
| private StreamReader srRules | Переменная для считывания символов из файла в кодировке ANSI. |

Таблица 5 – Поля визуальных элементов управления класса MainWindow

|  |  |
| --- | --- |
| Поле | Назначение |
| internal Window wndResult | Окно «Победа», в котором выдается результат игры. |
| internal Button btnNewGame | В окне «Победа» кнопка, для перемешивания значений и начала новой игры. |
| internal Button btnCloseGame | В окне "Победа" кнопка "Закрыть игру". |
| internal Label lblTimeResult | Метка в окне «Победа», в которую выводится результат игры (время в секундах). |

Таблица 6 – Собственные методы класса MainWindow

|  |  |
| --- | --- |
| Метод | Назначение |
| private void Timer\_Tick(object sender, EventArgs e) | Метод для обработки каждого тика, с момента запуска игры. Каждую секунду увеличивает значение переменной timer на 1 секунду. |

Таблица 7 – Методы класса MainWindow, обрабатывающие события

|  |  |
| --- | --- |
| Метод | Назначение |
| private void btn\_Click(object sender, RoutedEventArgs e) | Метод, обрабатывающий событие клика на кнопку внутри игрового поля, для перемещения её на соседнюю позицию, если там находится пустая клетка. Так же, метод совершает проверку, закончена ли игра. Если закончена, останавливает таймер и кладет в метку lblTimeResult время, за которое игра пройдена. |
| private void btnMix\_Click(object sender, RoutedEventArgs e) | Метод, обрабатывающий событие нажатия на кнопку «Начать игру» в основном окне и в окне «Победа». С помощью переменной из класса Random случайным образом перемешивает значения кнопок в области игрового поля. На основе математического описания вычисляет, решаема ли головоломка, если не решаема, то снова «перемешивает» значения. |
| private void miRulesOfGame\_Click(object sender, RoutedEventArgs e) | Метод, обрабатывающий событие нажатия на элемент в меню «Справка»-«Правила игры». Выдает MessageBox с сообщением об игре и правилах игры. |
| private void btnCloseGame\_Click(object sender, RoutedEventArgs e) | Метод, закрывающий главное окно, а также все открытые дочерние окна при нажатии на кнопку «Закрыть». |

Алгоритм работы с классом MainWindow:

1) Создается лист кнопок непосредственно игрового поля, каждая кнопка инициализируется значением по умолчанию (создаётся игровое поле, представленное в решенном виде). Каждой кнопке присваивается событие Click.

2) Динамически создается окно «Победа» с результатом игры.

3) После нажатия кнопки «Начать игру» запускается метод Timer\_Tick и игра начинается.

4) Игра завершается при условии, что кнопка «Начать игру» была нажата, и пользователь выполнил цель игры: упорядочить кнопки по их значениям по возрастанию. При выполнении условия появляется окно «Победа», в котором указано время, за которое пройдена игра.

# **4 Руководство пользователя**

## **4.1 Сведения о файлах программы и способе их установки**

Для использования программы нужно скопировать файл Fifteen.exe в любую папку файловой системы компьютера.

Требования к ресурсам компьютера:

* Наличие ОС Windows 7 и выше;
* Наличие Microsoft .NET Framework 4.5.2;
* Для корректной работы программы для всех устройств ЭВМ должны быть установлены драйвера;
* Разрешение экрана 800х600 и выше, 256 цветов;
* 1,25 мегабайт свободного дискового пространства;
* Процессор с тактовой частотой от 500 МГц;
* Манипулятор мышь – для выбора цифр, нажатия кнопок и многого другого;
* Монитор.

## **4.2 Интерфейс программы**

Программа запускается двойным кликом мыши по иконке с названием Fifteen.exe. На экране появится основное окно программы, содержащее в себе игровое поле, кнопки меню «Игра» и «Справка» и таймер (см. рисунок 1).



Рисунок 1 – Главное окно «Fifteen»

Меню содержит в себе элементы, изображенные на рисунке 2.

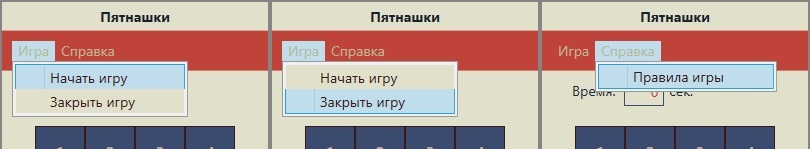


Рисунок 2 – Содержание меню программы

Для того что бы начать игру, пользователь должен нажать на кнопку в меню: «Игра» ‒ «Начать игру», после чего запустится таймер и игра начнется.

Игра заканчивается при достижении определенных условий. В таком случае появится уведомляющее окно, в котором будут показаны уведомление об окончании игры и время, за которое эта игра была пройдена (рисунок 3).

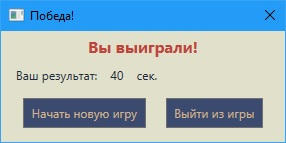


Рисунок 3 – Информирование о завершении игры

После окончания игры пользователь имеет право закончить игру, нажав на кнопку «Выйти из игры», после чего закроется главное окно и все дочерние окна. Или пользователь имеет право начать новую игру, нажав на кнопку «Начать новую игру», после чего сразу вновь перемешаются значения, запустится таймер и игра начнется.

Чтобы узнать дополнительные сведения об игре, пользователь должен нажать кнопку в меню: «Справка» ‒ «Правила игры» (рисунок 4).

Чтобы закрыть игру пользователь должен нажать кнопку в меню: «Игра» ‒ «Закрыть игру», после чего закроется главное окно и все дочерние окна.

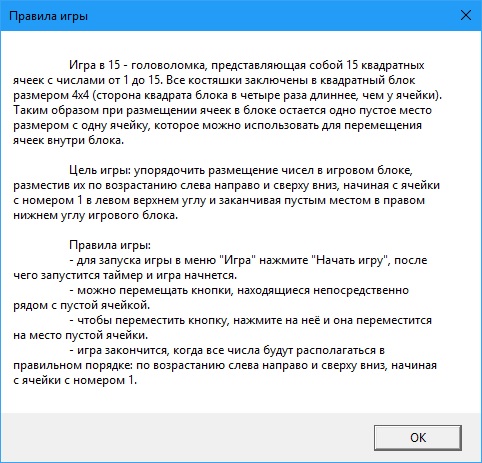


Рисунок 4 – Справка об игре

# **Заключение**

Для разработки данного программного средства были изучены научные статьи, посвященные математическому описанию головоломки «Игра в 15», и основные возможности Windows Presentation Foundation, такие как создание оконного приложения с графической составляющей и анимацией, создание удобного пользовательского интерфейса.

Разработанная программа может быть использована в целях развития логического мышления, усидчивости и способности к анализу, а также она имеет развлекательный характер.

В будущем планируется улучшение данной программы: доработка графического интерфейса и анимации, возможность менять цветовую палитру в связи с предпочтениями пользователя, сохранение статистики игр, добавление изображений, которые можно будет собирать, вместо цифр.

# **Список использованных источников**

1. Калужнин Л. А., Сущанский В. И. Преобразования и перестановки: Пер. с укр.—2-е изд., перераб. и доп. — М.: Наука. Главная редакция физико-математической литературы, 1985
2. Натан А. WPF 4. Подробное руководство. / А. Натан; пер. с англ. А. Слинкина – СПб.: Символ-Плюс, 2011. – 880 с.
3. Мозговой М. В., Занимательное программирование – СПб: Издательский дом "Питер", 2005 – 208 с.
4. Материалы сайта http://professorweb.ru
5. Материалы сайта http://habrahabr.ru
6. Материалы сайта http://msdn.microsoft.com

# **Приложение А**

***(обязательное)***

**Текст программы**

**MainWindow.xaml.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Threading;

using System.IO;

namespace Fifteen

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

List<Button> ListOfButtons = new List<Button>();

bool btnMixIsPressed = false;

// окно "Победа", в котором выдается результат

Window wndResult = new Window();

// в окне "Победа" кнопка "Начать новую игру"

Button btnNewGame = new Button();

// в окне "Победа" кнопка "Закрыть игру"

Button btnCloseGame = new Button();

// таймер для игры

DispatcherTimer timer = new DispatcherTimer();

// время, за которое игра будет пройдена

int myTime = 0;

// Label, в котором будет выдан результат

Label lblTimeResult = new Label();

string strRules; //текст с правилами игры

public MainWindow()

{

InitializeComponent();

int y = 1;

//создаем игровое поле с 4х4 с кнопками

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

{

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

{

ListOfButtons.Add(new Button()); //добавляем новую кнопку

ListOfButtons[ListOfButtons.Count - 1].Name = "b" + y.ToString(); //задаем имя кнопки

y++; //увеличиваем значение счетчика на 1

//устанавливаем позицию кнопки

ListOfButtons[ListOfButtons.Count - 1].Margin = new Thickness(50 \* i, 50 \* j, 50 \* (3 - i), 50 \* (3 - j));

gridMain.Children.Add(ListOfButtons[ListOfButtons.Count - 1]); //добавляем кнопку в поле gridMain

ListOfButtons[ListOfButtons.Count - 1].Background = new SolidColorBrush(Color.FromRgb(0x3a, 0x4b, 0x6f));

ListOfButtons[ListOfButtons.Count - 1].BorderBrush = new SolidColorBrush(Color.FromRgb(0x39, 0x18, 0x1a));

if (i != 3 || j != 3) //если это не последняя кнопка, то будем задавать ей значение по умолчанию

{

ListOfButtons[ListOfButtons.Count - 1].Content = (j \* 4 + i + 1); //кладём в кнопку значение по умолчанию

ListOfButtons[ListOfButtons.Count - 1].FontSize = 16;

ListOfButtons[ListOfButtons.Count - 1].FontWeight = FontWeights.Medium;

ListOfButtons[ListOfButtons.Count - 1].Foreground = new SolidColorBrush(Colors.Tan);

}

ListOfButtons[ListOfButtons.Count - 1].Click += btn\_Click; //присваивем кнопке метод клика

}

}

//окно с результатом

wndResult.Visibility = Visibility.Hidden; //по умолчанию делаем окно с результатом невидимым

wndResult.Width = 300;

wndResult.Height = 150;

wndResult.ResizeMode = ResizeMode.NoResize;

wndResult.WindowStartupLocation = WindowStartupLocation.CenterScreen;

wndResult.Title = "Победа!";

wndGame.Closing += new System.ComponentModel.CancelEventHandler(wndGameClosing);

StackPanel spResult = new StackPanel(); //кладём в окно с результом основной StackPanel

wndResult.Content = spResult;

spResult.Background = new SolidColorBrush(Color.FromRgb(0xe1, 0xe1, 0xcb));

StackPanel spResult1 = new StackPanel();

StackPanel spResult2 = new StackPanel();

StackPanel spResult3 = new StackPanel();

StackPanel spResult4 = new StackPanel();

spResult.Children.Add(spResult1);

spResult.Children.Add(spResult2);

spResult.Children.Add(spResult3);

spResult.Children.Add(spResult4);

// создаем Label "Вы выиграли"

Label lblYouAreWin = new Label();

lblYouAreWin.Content = "Вы выиграли!";

lblYouAreWin.Foreground = new SolidColorBrush(Color.FromRgb(0xc0, 0x43, 0x39));

lblYouAreWin.FontSize = 16;

lblYouAreWin.FontWeight = FontWeights.Bold;

spResult1.Children.Add(lblYouAreWin);

lblYouAreWin.HorizontalAlignment = HorizontalAlignment.Center;

spResult2.Orientation = Orientation.Horizontal;

// создаем Label "Ваш результат"

Label lblResult = new Label();

spResult2.Children.Add(lblResult);

lblResult.Content = " Ваш результат: ";

spResult2.Children.Add(lblTimeResult);

lblResult.Foreground = new SolidColorBrush(Color.FromRgb(0x1D, 0x29, 0x3E));

//устанавливаем таймер

timer.Interval = new TimeSpan(0, 0, 1);

timer.Tick += Timer\_Tick;

lblTimeResult.Foreground = new SolidColorBrush(Color.FromRgb(0x1D, 0x29, 0x3E));

// создаем Label "сек."

Label lblSec = new Label();

lblSec.Content = " сек.";

spResult2.Children.Add(lblSec);

lblSec.Foreground = new SolidColorBrush(Color.FromRgb(0x1D, 0x29, 0x3E));

// StackPanel с кнопками "Начать новую игру" и "Выйти из игры"

spResult4.Orientation = Orientation.Horizontal;

spResult4.HorizontalAlignment = HorizontalAlignment.Center;

btnNewGame.Margin = new Thickness(10);

btnNewGame.Content = " Начать новую игру ";

btnNewGame.Foreground = new SolidColorBrush(Colors.Tan);

btnNewGame.Background = new SolidColorBrush(Color.FromRgb(0x3a, 0x4b, 0x6f));

btnNewGame.Height = 30;

btnNewGame.Click += btnMix\_Click;

btnCloseGame.Content = " Выйти из игры ";

btnCloseGame.Foreground = new SolidColorBrush(Colors.Tan);

btnCloseGame.Background = new SolidColorBrush(Color.FromRgb(0x3a, 0x4b, 0x6f));

btnCloseGame.Margin = new Thickness(10);

btnCloseGame.Click += btnCloseGame\_Click;

btnCloseGame.Height = 30;

spResult4.Children.Add(btnNewGame);

spResult4.Children.Add(btnCloseGame);

//окно "Правила игры"

FileStream fsRules = new FileStream("Rules.txt", FileMode.Open, FileAccess.Read);

StreamReader srRules = new StreamReader(fsRules, Encoding.Default);

strRules = srRules.ReadToEnd();

srRules.Close();

fsRules.Close();

}

private void Timer\_Tick(object sender, EventArgs e)

{

myTime++; //за один тик увеличиваем счетчик времени - увеличиваем время на 1 секунду

lblTimer.Content = myTime; // кладём значение в Label со временем в окне с игрой

}

// обрабатываем событие клика по кнопке внутри блока с самой игрой

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

{

//проверка, являются ли кнопки соседними через значения Margin

if (((Math.Abs((sender as Button).Margin.Left - ListOfButtons[ListOfButtons.Count - 1].Margin.Left) + Math.Abs((sender as Button).Margin.Top - ListOfButtons[ListOfButtons.Count - 1].Margin.Top))) == 50)

{

Thickness th = new Thickness();

th = ListOfButtons[ListOfButtons.Count - 1].Margin;

ListOfButtons[ListOfButtons.Count - 1].Margin = (sender as Button).Margin;

(sender as Button).Margin = th;

}

// проверка, решена ли головоломка

if (btnMixIsPressed) // проверка, нажата ли кнопка "Начать игру"

{

int l = 0; //счетчик ячеек, совпадающих с их положением в решенной головоломке

for (int i = 0; i < 15; i++) //счетчик всех ячеек

{

int marginLeft = int.Parse(ListOfButtons[i].Margin.Left.ToString()); //находим значение Margin от левого края

marginLeft = marginLeft / 50 + 1; //находим позицию ячейки по горизонтали

int marginTop = int.Parse(ListOfButtons[i].Margin.Top.ToString()); //находим значение Margin от верхнего края

marginTop = (marginTop / 50) \* 4; //находим позицию ячейки по вертикали

//если значение сумм Margin(положение кнопки с цифрой в блоке игры) равно содержимому кнопки, то увеличиваем счетчик l

if ((marginLeft + marginTop) == int.Parse(ListOfButtons[i].Content.ToString()))

{

l++;

}

}

//проверяем счетчик

if (l == 15)

{

//если каждая кнопка оказывается на своём месте, то делаем окно с результатом видимым

wndResult.Visibility = Visibility.Visible;

timer.Stop(); //останавливаем таймер

lblTimeResult.Content = myTime; //в Label с результатом кладём значение времени

}

}

}

//кнопка для перемешивания значений в ячейках для начала игры

private void btnMix\_Click(object sender, RoutedEventArgs e) //перемешать

{

myTime = 0; //обнуляем таймер в начале игры

timer.Start(); //запускаем таймер

wndResult.Visibility = Visibility.Hidden; //делаем окно "Победа" невидимым

btnMixIsPressed = true; //устанавливаем флаг, что игра начата

int N; //сумма, для определения, решаема ли головоломка

//если N - нечетная, то решения головоломки не существует

do

{

N = 0;

Random random = new Random();

for (int i = 0; i < 15 / 2; i++)

{

int i1 = random.Next(0, 15); // первый индекс

int i2 = random.Next(0, 15); // второй индекс

// обмен значений элементов с индексами i1 и i2

int temp = int.Parse(ListOfButtons[i1].Content.ToString());

ListOfButtons[i1].Content = ListOfButtons[i2].Content;

ListOfButtons[i2].Content = temp;

}

//находим значение N, для того что бы определить, решаема ли головоломка (это берем из её метематического описания)

for (int j = 1; j < 15; j++) //начинаем со второй ячейки (с индексом 1)

{

int k = 0; //счетчик количества ячеек, имеющих значение больше, чем j, но находящихся перед ней в таблице головоломки

//изначально обнуляем его

for (int i = 0; i < j; i++) //проходим все ячейки, находящиеся перед ячейкой с индексом j

{

//сравниваем значения лежащие в ячейках с индексами i и j, если в i-ой ячейке находится значение больше, чем в j-ой, то увеличиваем счетчик

if (int.Parse(ListOfButtons[j].Content.ToString()) < int.Parse((ListOfButtons[i].Content.ToString())))

{

k++; //увеличиваем счетчик

}

}

N += k; //находим сумму количества таких ячеек

}

int row = (int.Parse(ListOfButtons[14].Margin.Top.ToString())) / 50 + 1; //находим ряд, в котором находится пустая ячейка

N += row;

} while (N % 2 == 1); //пока сумма нечетна, перемешиваем, пока N ни станет четным и головоломка ни станет решаемой

}

//элемент из меню, показывающий MessageBox с информацией об игре

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

{

MessageBox.Show(strRules, "Правила игры", MessageBoxButton.OK);

}

//кнопка закрытия игры в окне с результатом

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

{

timer.Stop();

wndResult.Close(); //закрываем окно результатов

wndGame.Close(); //закрываем окно с игрой

}

private void wndGameClosing(object sender, EventArgs e)

{

wndResult.Close(); //закрывание окна результатов

}

}

}