**Федеральное государственное образовательное бюджетное учреждение  
высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ**

**ПМ.03 Участие в интеграции программных модулей**

**Группа: 4ПКС-115**

**УТВЕРЖДАЮ**

**Председатель цикловой комиссии**

**программирования и баз данных**

**( ) Пестов А.И.**

**\_\_\_\_.\_\_\_\_. 2018**

**ПРОЕКТ КУРСОВОЙ**

**На тему: Разработка компьютерной игры с графическим интерфейсом «Сапёр» на языке программирования С#**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**Руководитель курсового проекта**

**(Морозова М.В.) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Исполнитель курсового проекта**

**(Черников А.В.) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Оценка за проект: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_.\_\_\_\_.2018**

**2018**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc532912776)

[1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 4](#_Toc532912777)

[1.1 Предпроектное исследование предметной области 4](#_Toc532912778)

[1.2 Характеристика системы программирования 7](#_Toc532912779)

[1.3 Анализ требований и определение спецификаций программного обеспечения 9](#_Toc532912780)

[2 СПЕЦИАЛЬНЫЙ РАЗДЕЛ 14](#_Toc532912781)

[2.1 Проектирование программного обеспечения 14](#_Toc532912782)

[2.2 Разработка базы данных 15](#_Toc532912783)

[2.3 Разработка справки 16](#_Toc532912784)

[2.4 Разработка пользовательских интерфейсов 18](#_Toc532912785)

[3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 20](#_Toc532912786)

[3.1 Технология связи с базой данных 20](#_Toc532912787)

[3.2 Технология связи со справкой 22](#_Toc532912788)

[3.3 Тестирование и отладка программы 22](#_Toc532912789)

[4 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ 25](#_Toc532912790)

[4.1 Руководство системного программиста 25](#_Toc532912791)

[4.2 Руководство программиста 27](#_Toc532912792)

[4.3 Руководство пользователя 32](#_Toc532912793)

[ЗАКЛЮЧЕНИЕ 53](#_Toc532912794)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc532912795)

[ПРИЛОЖЕНИЕ А 56](#_Toc532912796)

[ПРИЛОЖЕНИЕ Б 57](#_Toc532912797)

[ПРИЛОЖЕНИЕ В 75](#_Toc532912798)

# ВВЕДЕНИЕ

Компьютерные технологии – детище научно-технического прогресса, набравшая к концу 20 века необыкновенно высокую скорость развития. Сложно недооценивать роль компьютеров в развитии общества. Компьютеры проникли во все сферы деятельности человека, начиная с начального образования и заканчивая изучением новейших технологий.

На работе, дома, в машине, да что уж там, даже в самолете мы тесно взаимодействуют с компьютером. Люди уже не замечают, насколько стремительно компьютеры внедряются в нашу жизнь и занимают, настолько важное место, что мы становимся буквально зависимыми от этих кусков металла.

Рассмотрим компьютерные игры. С древних времен люди увлекались играми, но в те далекие времена эти самые игры были не такими уж мирными. В современном мире это увлечение никуда не ушло, а вышло на новый уровень.

В наше время их настолько много, что в сети «Интернет» можно найти немало материала по ним. Мир виртуальных развлечений разнообразен и интересен и здесь каждый сможет найти то, что ему нравится. Основную массу людей привлекает возможность сделать то, что в реальной жизни сделать крайне сложно. Так, в виртуальном мире вы можете стать космонавтом, успешным спортсменом, волшебником и так далее.

Понятие «компьютерные игры» вошло в нашу жизнь сравнительно недавно, но уже каждый обладатель компьютера смог испытать их и проникнуться ими.

Данный проект посвящен разработке компьютерной игры «Сапёр», в котором будет реализован аналог этой классической игры с добавлением некоторых новшеств.

# ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

## Предпроектное исследование предметной области

### Описательная модель предметной области

Данный проект был разработан в целях обучение пользователя быстрому и точному использованию левой и правой кнопок мыши, а также развитие логического мышления посредством игры. Перед тем как приступить к созданию программы были изучены подобные аналоги. В каждом из них было достаточно много функций среди которых удалось выделись наиболее часто используемые и на их основании составить описательную модель.

Программа предназначена для владельцев персонального компьютера (ПК).

Необходимо разработать компьютерную игру «Сапёр». Цель игры состоит в том, чтобы как можно быстрее найти все мины на минном поле, не вскрыв ни одну из них. Игрок имеет возможность открыть любую ячейку. Если открываемая ячейка содержит мину игра заканчивается проигрышем. Если мины нет, в ячейке появится цифра, которая указывает, сколько мин находится в восьми смежных с ней ячейках. Игра заканчивается выигрышем, как только игрок откроет все клетки, не содержащие мин.

В игре необходимо предусмотреть:

* различные режимы (отличающиеся количеством мин, шириной и высотой);
* таблицу рекордов;
* возможность установки флажков (нажатием на правую кнопку мыши);
* таймер;
* ограниченное количество флажков (не больше, чем мин на поле);
* изменение «смайлика» в зависимости от результатов игры (нейтральный (во время игры), в очках (при победе), мёртвый (при поражении));
* «горячие» клавиши для каждой основной функции программы;
* справку (содержащую инструкцию к игре и информацию об авторе).

### Описание входной информации

Входные данные – данные вводимые или выбранные пользователем в ходе работы программы. К ним относится ввод имени в таблицу рекордов, а также ручная установка размеров игрового поля и количества мин на нём.

### Описание выходной информации

Выходные данные – данные, полученные в ходе работы пользователя с программой. К ним относится запись введённого пользователем имени в базу данных (БД), функции вывода игрового поля на экран монитора, количество оставшихся на поле мин и время на таймере, прошедшее с начала игры.

### Требования к программному обеспечению

#### Системные требования:

Необходимыми программными средствами являются:

* операционная система Windows 7/8/10;
* наличие .NET Framework версии 4.6.

Необходимыми аппаратными средствами для работы приложения являются:

* процессор с тактовой частотой 2,6 ГГц или выше;
* 6 ГБ ОЗУ (8 ГБ при выполнении на виртуальной машине);
* монитор;
* компьютерная мышь либо наличие touchpad;
* клавиатура.

#### Функциональные и нефункциональные требования:

Функциональные:

* работать с заданным алгоритмом функционирования;
* поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей;
* производить бесперебойную работу по преобразованию информации.

Нефункциональные:

* интерфейс пользователя;
* аппаратный интерфейс.

#### Требования к реализации:

Программа должна быть написана на языке программирования C# с использованием объектно-ориентированного программирования (ООП).

#### Требования к надёжности

В программном продукте предусмотрена защита от действий пользователя, таких как:

* ввод значений размеров игрового поля превышающих допустимые;
* ввод некорректного имени для таблицы рекордов;
* защита БД рекордов.

#### Требования к интерфейсу:

Интерфейс программы должен быть простой и понятный для пользователя, выводить подсказки к некоторым элементам, помогать выполнять основные функции быстрее и с наименьшими усилиями и привлекательно смотреться.

#### Требования к программной документации:

Программная документация оформляется в соответствии с требованиями документа «МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОФОРМЛЕНИЮ КУРСОВОЙ И ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ» и указанными в нём ГОСТами.

## Характеристика системы программирования

### Характеристика среды программирования

При работе над курсовым проектом использовалась среда программирования «Microsoft Visual Studio 2017», представляющая собой полный набор средств разработки для создания веб-приложений ASP.NET, XML (веб-службы), настольных приложений и мобильных приложений. «Visual Studio» использует единую интегрированную среду разработки (IDE), которая позволяет совместно использовать средства и упрощает создание решений на базе нескольких языков. Кроме того, в этих языках используются функциональные возможности платформы .NET Framework, которая позволяет получить доступ к ключевым технологиям, упрощающим разработку веб-приложений ASP и XML.

«Microsoft Visual Studio» объединяет в себе огромное количество функций, позволяющих осуществлять разработки для Windows всех версий, Интернета, SharePoint, различных мобильных устройств и облачных технологий. В Visual Studio реализуется новая среда разработчика, благодаря которой создавать приложения стало проще.

### Описание языка программирования

В качестве языка программирования был выбран язык С#.

**Язык C#** появился на свет в июне 2000 г., в результате кропотливой работы большой группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом*.*

Создание инструментария для разработчиков с их полноценной поддержкой является одной из главных задач нового языка C#*.*

**Авторы C#** стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков c богатством возможностей и мощью C++.

Особенности С#:

* полная поддержка классов и объектно-ориентированного программирования, включая наследование интерфейсов и реализаций, виртуальных функций и перегрузки операторов;
* полный и хорошо определенный набор основных типов;
* встроенная поддержка автоматической генерации XML-документации;
* возможность отметки классов и методов атрибутами, определяемыми пользователем. Это может быть полезно при документировании и способно воздействовать на процесс компиляции;
* автоматическое освобождение динамически распределенной памяти;
* полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows API;
* указатели и прямой доступ к памяти, если они необходимы. Однако язык разработан таким образом, что практически во всех случаях можно обойтись и без этого;
* поддержка свойств и событий в стиле VB;
* простое изменение ключей компиляции. Позволяет получать исполняемые файлы или библиотеки компонентов .NET, которые могут быть вызваны другим кодом так же, как элементы управления ActiveX;
* возможность использования С# для написания динамических web-страниц ASP.NET.

### Описание системы Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) — это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность.

В основе WPF лежит независимый от разрешения векторный модуль визуализации, использующий возможности современного графического оборудования. Возможности этого модуля расширяются с помощью комплексного набора функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF входит в состав .NET Framework, поэтому вы можете создавать приложения, включающие другие элементы библиотеки классов .NET Framework.

При разработке поведения приложения главной задачей является обеспечение реакции на действия пользователя, включая обработку событий (таких как выбор пункта меню или нажатие на кнопку), и вызов в ответ бизнес-логики и логики доступа к данным. В WPF такое поведение реализуется в коде, связанном с разметкой. Этот код называется кодом программной части.

Возможности взаимодействия с пользователем, обеспечиваемые моделью приложения, реализуются с помощью сконструированных элементов управления. В WPF элемент управления — это общий термин, который относится к категории классов WPF, размещаемых в окне или на странице, имеющих пользовательский интерфейс и реализующих некоторое поведение.

## Анализ требований и определение спецификаций программного обеспечения

### Диаграмма вариантов использования

Диаграмма вариантов использования представлена на рисунке 1.1. Она описывает взаимоотношения и зависимости между группами вариантов использования и игроком, участвующими в процессе.

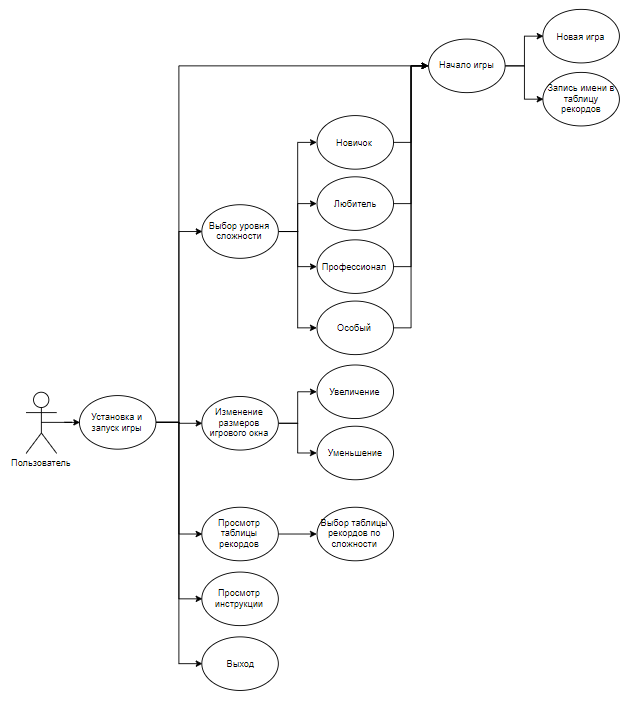


Рисунок 1.1 – Диаграмма вариантов использования

### Диаграмма потоков данных

Контекстная диаграмма потоков данных представлена на рисунке 1.2. Она представляет собой взаимодействие пользователя и приложения, связанных потоками данных.

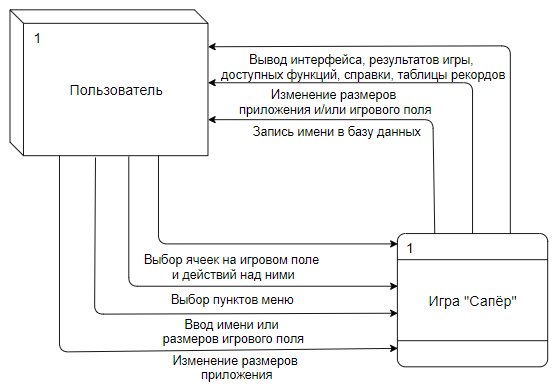


Рисунок 1.2 – Контекстная диаграмма потоков данных

Детализированная диаграмма потоков данных представлена на рисунке 1.3, она является более подробной версией контекстной диаграммы потоков данных.

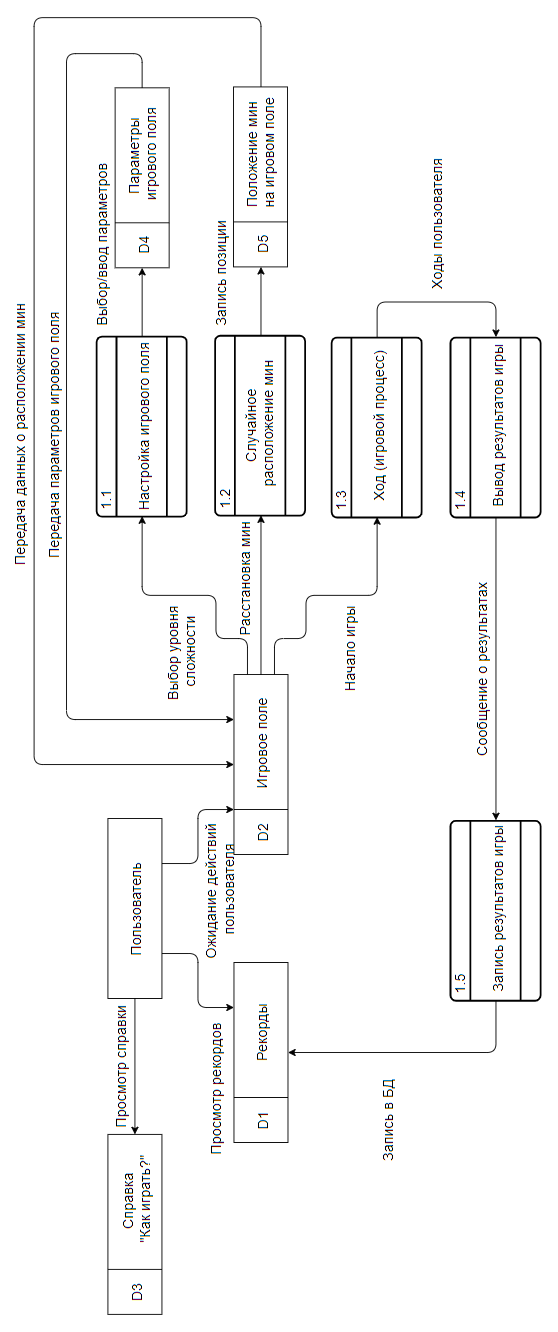


Рисунок 1.3 – Детализированная диаграмма потоков данных

### Функциональная диаграмма

Функциональная диаграмма – диаграмма, отражающая взаимосвязи функций разрабатываемого программного обеспечения (ПО).

Контекстная функциональная диаграмма представлена на рисунке 1.4.

На ней видно, что на вход программе подаются выбранные или введённые пользователем данные (выбор уровня сложности и/или самостоятельный ввод ширины, высоты и количества мин). Программа опирается на правила игры «Сапёр». Программой управляет пользователь. На выходе у программы результаты игры.

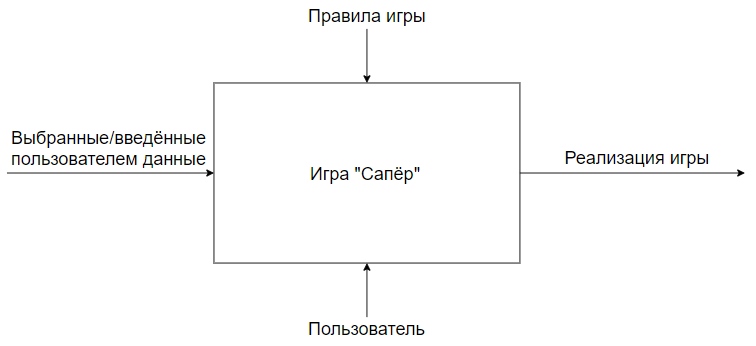


Рисунок 1.4 – Контекстная функциональная диаграмма

Детализированная функциональная диаграмма представлена на рисунке 1.5.

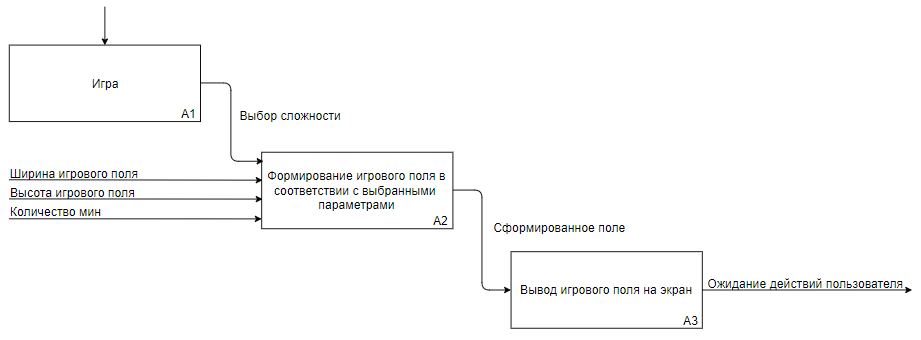


Рисунок 1.5 – Детализированная функциональная диаграмма

# СПЕЦИАЛЬНЫЙ РАЗДЕЛ

## Проектирование программного обеспечения

Структурная схема приведена на рисунке 2.1. Она демонстрирует последовательность выполнения действий, реализующих принцип функционирования программы.

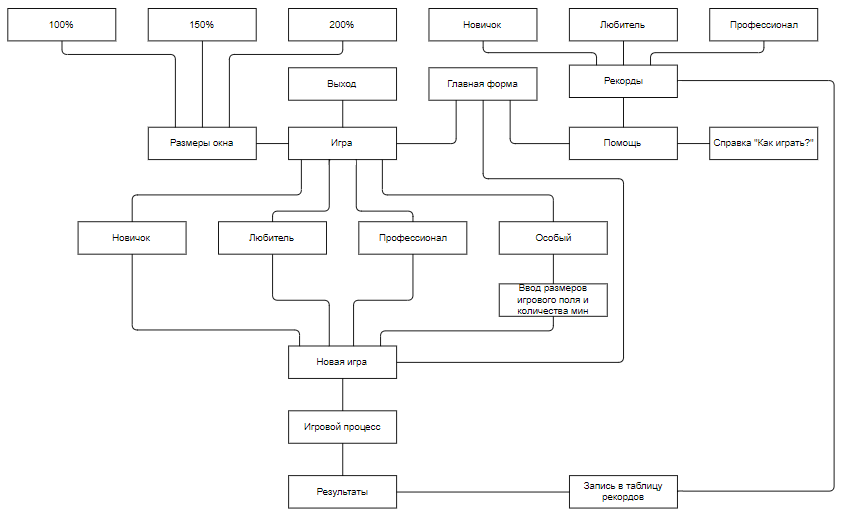


Рисунок 2.1 – Структурная схема

Функциональная схема приведена на рисунке 2.2. Она разъясняет процессы, протекающие в отдельных функциях программы и программы в целом.

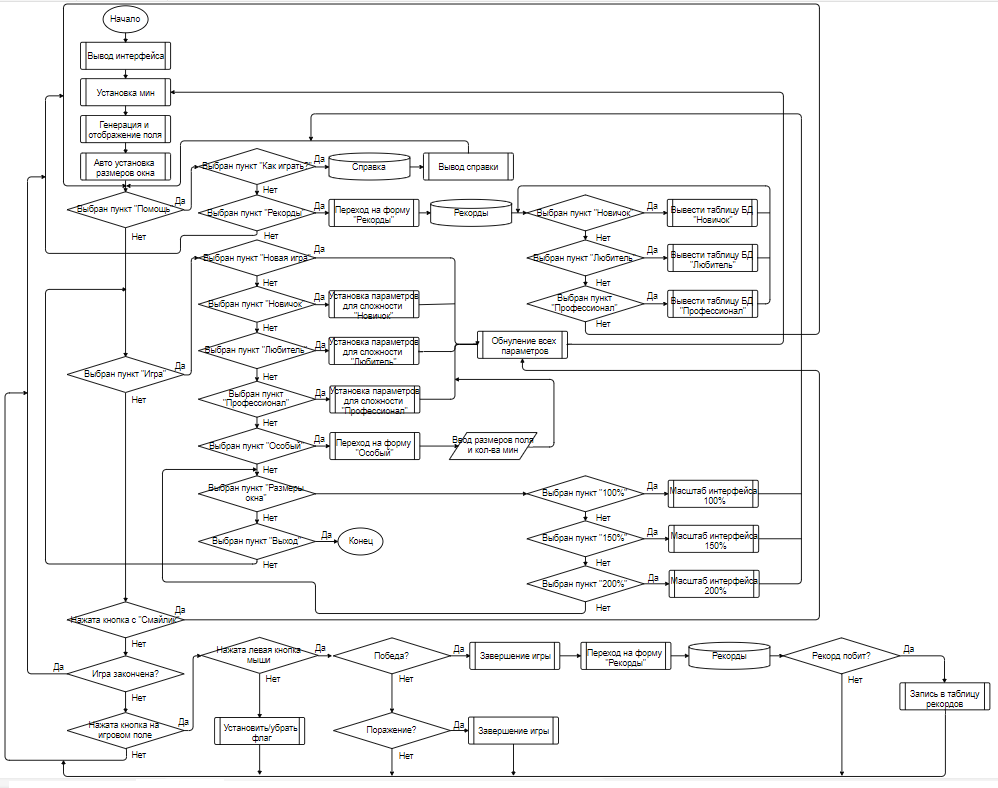


Рисунок 2.2 – Функциональная схема

## Разработка базы данных

Входе работы над курсовым проектом возникла необходимость в создании БД для хранения рекордов лучших игроков. В качестве системы управления БД была выбрана SQLite.

Спроектированная БД имеет всего 3 таблицы: «Beginner», «Enthusiast», «Professional», они представлены на рисунке 2.3. Каждая из них соответствует одному из уровней сложности игры, а именно «Новичок», «Любитель», «Профессионал». Таблица для уровня сложности «Особый» не была включена в БД, так как в этом режиме игрок сам задаёт размер поля и количество мин, в связи с чем отслеживание результата не имеет смысла.

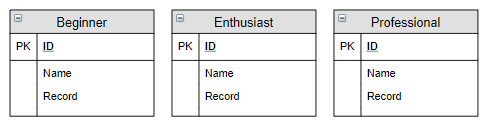


Рисунок 2.3 – Схема базы данных

Таблицы не связаны между собой, так как каждая из них отвечает за свой уровень сложности. Они имеют следующие поля:

* «ID»;
* «Name»;
* «Record».

Поле «ID» отвечает за порядковый номер игрока в списке рекордов. Оно является уникальным и не может превышать 10 пунктов.

Поле «Name» отвечает за имя игрока, который сумел побить рекорд другого игрока или игроков. Оно не может содержать пробелы и превышать длину 25 символов.

Поле «Record» отвечает за рекорд игрока, то время, за которое он смог победить в игре. Именно по этому полю порядок игроков в таблицах. Если рекорд игроков совпал, тогда игрок, получивший этот рекорд позже ставится выше игрока, который получил этот рекорд раньше.

Изначально в каждой таблице БД прописаны 10 игроков с их рекордами, чтобы пользователь знал к чему стоит стремиться.

## Разработка справки

Входе работы над курсовым проектом возникла необходимость в создании справки (инструкции). Для её создания была выбрана программа htm2chm.

Думаю, многие из вас сталкивались с такой проблемой. Скачиваешь программу – видишь кучу полезной информацию, но при этом не можешь разобраться как полностью использовать все возможности программы, а пользоваться каждый раз Google, мягко говоря, напрягает. Да и плюс ко всему хотелось бы иметь возможность иметь информацию под другой даже когда нет интернета.

Справка, созданная специально для этого курсового проекта, имеет следующие разделы:

* «Содержание»;
* «Вступление»;
* «Настройки»;
* «Как играть?»;
* «Заключение»;
* «Информацию об авторе»;

Каждый из разделов, о которых более подробно будет рассказано позднее, был написан на языке вёрстки HTML. В связи с этим справка получилось достаточно простой, но в тоже время информативной. На рисунке 2.4 показаны файлы, каждый из которых отвечает за определённый раздел справки.

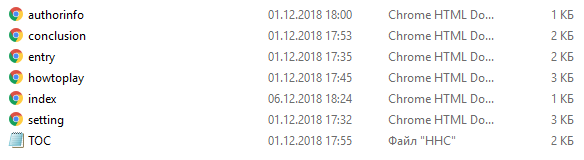


Рисунок 2.4 – Файлы, используемые в справке

Используемые файлы:

* «authorinfo» – отвечает за «Информацию об авторе»;
* «conclusion» – отвечает за «Заключение»;
* «entry» – отвечает за «Вступление»;
* «howtoplay» – отвечает за «Как играть?»;
* «index» – отвечает за «Содержание»;
* «setting» – отвечает за «Настройки»;
* «ТОС» – отвечает за содержание справки в левой части окна справки, чтобы при необходимости из любого раздела можно было быстро перейти в любой другой.

## Разработка пользовательских интерфейсов

При запуске программы перед пользователем появляется главная форма (рисунок 2.5).

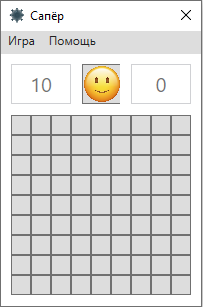


Рисунок 2.5 – Главное окно программы

Из этого окна доступно большинство функций программы включая и сам игровой процесс (рисунок 2.6).

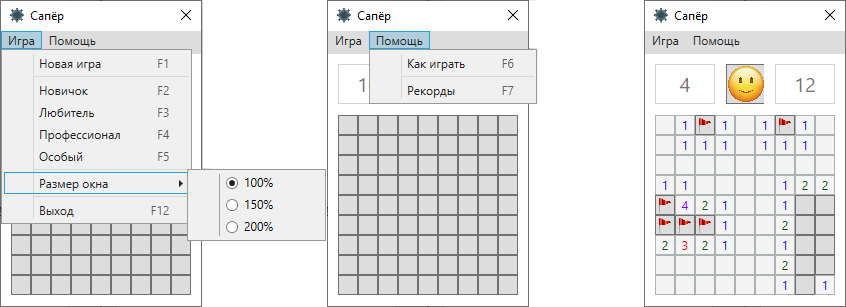


Рисунок 2.6 – Большинство функций программы

Справка продемонстрирована на рисунке 2.7.

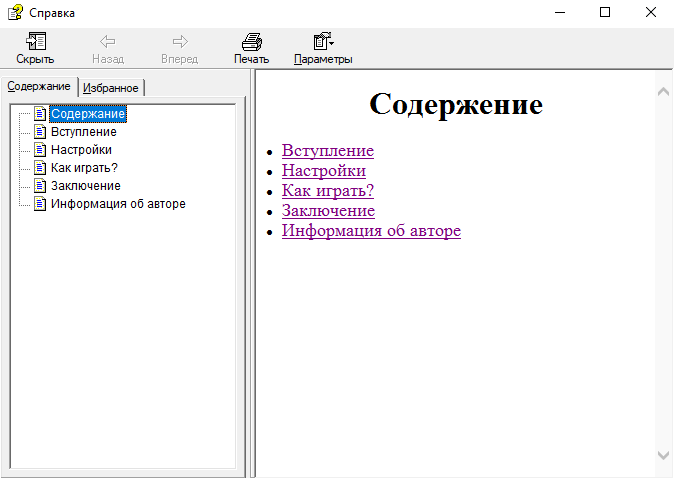


Рисунок 2.7 – Внешний вид справки

Форма пункта «Рекорды» продемонстрирована на рисунке 2.8.

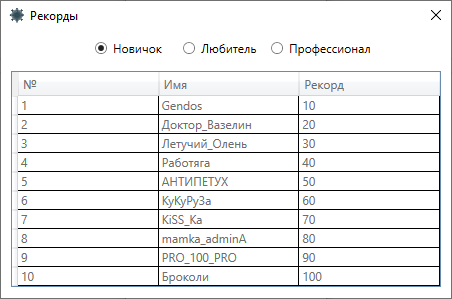


Рисунок 2.8 – Окно «Рекорды»

# ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## Технология связи с базой данных

Сегодня большое значение имеет работа с данными. Для хранения данных используются различные системы управления БД: MS SQL Server, SQLite, MySQL и так далее. И большинство крупных приложений так или иначе используют для хранения данных эти системы управления БД. Однако, чтобы осуществлять связь между БД и приложением на C# необходим посредник. И именно таким посредником является технология ADO.NET.

ADO.NET (ActiveX Data Object для .NET) – технология, предоставляющая доступ и управление данными, хранящимся в БД или других источниках, основанных на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к БД, устанавливать подключения, получать ответ от БД и производить ряд других операций.

В отличие от технологии ADO, которая была в основном предназначена для тесно связанных клиент-серверных систем, ADO.NET больше нацелена на автономную работу с помощью объектов DataSet. Объекты DataSet представляют локальные копии взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Объекты DataSet позволяют вызывающей сборке (наподобие веб-страницы или программы, выполняющейся на настольном компьютере) работать с содержимым DataSet, изменять его, не требуя подключения к источнику данных, и отправлять обратно блоки измененных данных для обработки с помощью соответствующего адаптера данных.

Причем важно отметить, что систем управления БД может быть множество. К примеру, в данном курсовом проекте была использована SQLite. Разные системы БД могут иметь разные типы данных. Также могут различаться какие-то другие моменты. Однако функционал ADO.NET построен таким образом, чтобы предоставить разработчикам унифицированный интерфейс для работы с самыми различными СУБД.

Основу интерфейса взаимодействия с БД в ADO.NET представляет ограниченный круг объектов: Connection, Command, DataReader, DataSet и DataAdapter.

С помощью объекта Connection происходит установка подключения к источнику данных.

Объект Command позволяет выполнять операции с данными из БД.

Объект DataReader считывает полученные в результате запроса данные.

Объект DataSet предназначен для хранения данных из БД и позволяет работать с ними независимо от БД. И объект DataAdapter является посредником между DataSet и источником данных.

Главным образом, через эти объекты и будет идти работа с БД.

Данный отрывок кода отвечает за подключение и выгрузку БД в программу:

*data = null;*

*FullPath = "Data Source = Records.db;";*

*SQLiteConnection connection = new SQLiteConnection(FullPath);*

*connection.Open();*

*string sql = "select \* from " + RecGameMode + ";";*

*SQLiteCommand command = new SQLiteCommand(sql, connection);*

*SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(command);*

*data = new DataTable(RecGameMode);*

*dataAdapter.Fill(data);*

*data.Columns["ID"].ColumnName = "№";*

*data.Columns["Name"].ColumnName = "Имя";*

*data.Columns["Record"].ColumnName = "Рекорд";*

*Table\_Records.ItemsSource = data.DefaultView;*

Первая строка обнуляет содержимое таблицы, содержащихся в объекте DataSet и представленных типом DataTable. Далее указывается название БД, происходит подключение, открытие соответствующей таблицы БД и выгрузка из неё нового содержимого в обнуляемый ранее объект DataSet. В последней строке наша таблица с новым содержимым выводится в элемент типа DataGrid.

## Технология связи со справкой

Как уже говорилось ранее, чтобы не тратить времени на поиски информации о том, как пользоваться программой в интернете или суметь ей воспользоваться даже, если интернета нет для этого курсового проекта была создана справка.

Её связь с программой осуществляется посредством следующего кода:

*string commandText = "Help.chm";*

*var proc = new System.Diagnostics.Process();*

*proc.StartInfo.FileName = commandText;*

*proc.StartInfo.UseShellExecute = true;*

*proc.Start();*

В первой строке указано название файла справки. Далее объявляется переменная класса «Process», этот класс предоставляет доступ к локальным и удалённым процессам и позволяет запускать и останавливать локальные системные процессы. Следующие 2 строки передают название файла, который должен быть открыт и указывают, должна ли использоваться оболочка операционной системы (ОС) при запуске процесса. Последняя строка запускает сам файл справки.

## Тестирование и отладка программы

В проекте существует пара моментов, где присутствует возможность возникновения ошибки, влекущей за собой дальнейшую неправильность работы программы. Это может произойти если что-нибудь случиться с файлом справки или с БД. На рисунках 3.1-3.4 продемонстрирована реакция программы на возникновение этих ошибок.

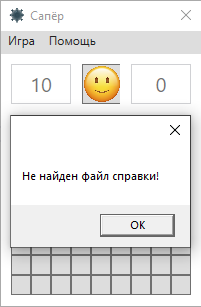


Рисунок 3.1 – Реакция программы на возникновение ошибки, связанной с файлом справки

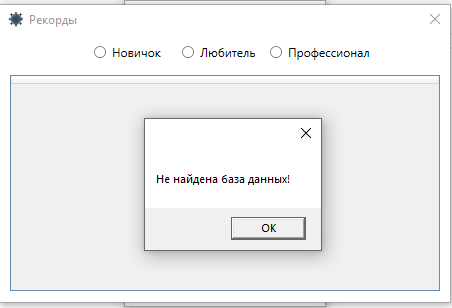


Рисунок 3.2 – Реакция программы на возникновение ошибки, связанной с БД

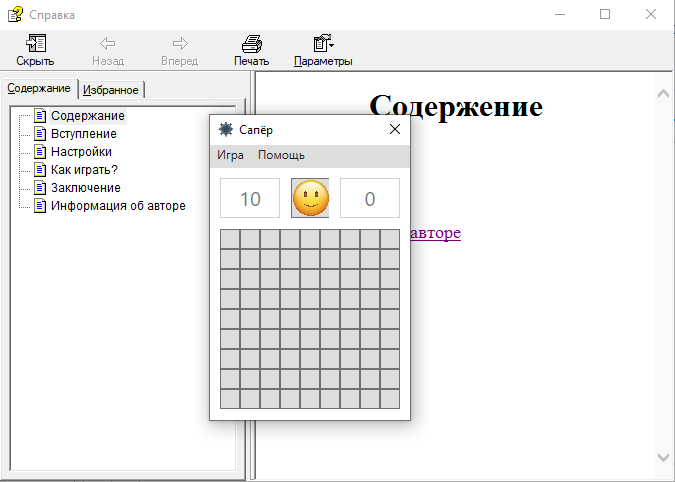


Рисунок 3.3 – Реакция программы при отсутствии ошибки, связанной с файлом справки

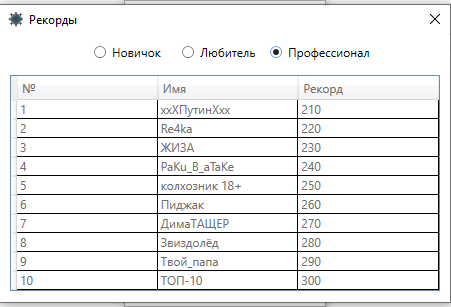


Рисунок 3.4 – Реакция программы при отсутствии ошибки, связанной с БД

# РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

## Руководство системного программиста

### Общие сведения о программе

Данная программа предназначена для обучения пользователя быстрому и точному использованию левой и правой кнопок мыши, а также развитие логического мышления посредством игры.

Применяя эту программу, пользователи имеют следующие возможности:

* игра на разных уровнях сложности;
* собственноручное создание игры с желаемыми параметрами;
* увеличение или уменьшение размеров интерфейса;
* просмотр существующих рекордов;
* внесение своего имени в таблицу рекордов;
* просмотр справки к игре.

### Файлы программы

Приложение состоит из следующих файлов:

* «S1WP.exe»;
* «Records.db»;
* «Help.chm»;
* «EntityFramework.dll»;
* «EntityFramework.SqlServer.dll»;
* «EntityFramework.SqlServer»;
* «EntityFramework»;
* «S1WP.exe.config»;
* «S1WP.pdb»;
* «System.Data.SQLite.dll»;
* «System.Data.SQLite.dll.config»;
* «System.Data.SQLite.EF6.dll»;
* «System.Data.SQLite.Linq.dll»;
* «System.Data.SQLite».

### Системные требования

Необходимыми системными требованиями являются:

* процессор с тактовой частотой 2,6 ГГц или выше;
* 6 ГБ ОЗУ (8 ГБ при выполнении на виртуальной машине);
* монитор;
* компьютерная мышь либо наличие touchpad;
* клавиатура.

### Программные требования

Необходимыми программными средствами являются:

* операционная система Windows 7/8/10;
* наличие .NET Framework версии 4.6.

### Настройка программы

Для установки программы необходимо распаковать архив «S1WP» (рисунок 4.1).

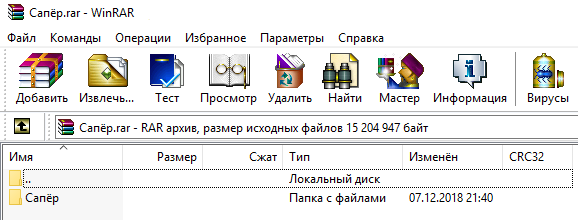


Рисунок 4.1 – Содержимое архива для установки программы

После завершение распаковки архива приложение будет готово к работе. Чтобы его запустить необходимо двойным щелчком левой кнопки мыши кликнуть по иконке с названием «S1WP.exe».

### Сообщения системному программисту

В данном курсовом проекте нет сообщений, предназначенных системному программисту.

## Руководство программиста

### Назначение и условия применения программы

Данная программа предназначена для обучения пользователя быстрому и точному использованию левой и правой кнопок мыши, а также развитие логического мышления посредством игры.

Необходимыми программными средствами являются:

* операционная система Windows 7/8/10;
* наличие .NET Framework версии 4.6.

Необходимыми аппаратными средствами для работы приложения являются:

* процессор с тактовой частотой 2,6 ГГц или выше;
* 6 ГБ ОЗУ (8 ГБ при выполнении на виртуальной машине)
* монитор;
* компьютерная мышь либо наличие touchpad;
* клавиатура.

### Структура программы

Таблица 4.1 – Модуль «MainWindow»

|  |  |
| --- | --- |
| Функция | Действие |
| Installation\_Mine() | Установка мин на игровом поле случайным образом, обнуление некоторых параметров |
| Creation\_Minefield() | Генерация и вывод на экран игрового поля, создание событий для левой и правой кнопок мыши каждой ячейке игрового поля |

Продолжение таблицы 4.1

|  |  |
| --- | --- |
| rightbutton\_Click() | Установка флажка на ячейку при нажатии на нею правой кнопкой мыши |
| button\_Click() | Открытие ячейки при нажатии на неё левой кнопкой мыши, проверка победы поражения и вывод результатов |
| SetButton() | Установка определённого изображения на нажатые ячейки |
| Step() | Определение нажатой ячейки, отображение цифр на ней и соседних ячейках и открытие пустых ячеек |
| ContainsIndex() | Помощь в определении положения ячейки |
| timer\_tick() | Увеличение значения таймера и его отображение |
| New\_Game() | Начало новой игры, обнуление некоторых параметров |
| Window\_Size() | Установка размеров окна и интерфейса |
| Beginner\_Mode() | Параметры для уровня сложности «Новичок» и начало новой игры |
| Enthusiast\_Mode() | Параметры для уровня сложности «Любитель» и начало новой игры |
| Professional\_Mode() | Параметры для уровня сложности «Профессионал» и начало новой игры |
| Transition\_Records\_Window() | Переход на форму «Рекорды» |
| Transition\_Special\_Window() | Переход на форму «Особый» |
| MineField\_Elements\_Size() | Размеры ячеек игрового поля |
| Game\_Size\_Switch() | Отслеживание размеров окна и интерфейса |

Продолжение таблицы 4.1

|  |  |
| --- | --- |
| Help() | Открытие справки |
| NewGame\_Click() | Начало новой игры, обнуление некоторых параметров при нажатии на определённую кнопку |
| Help\_Click() | Открытие справки при нажатии на определённую кнопку |
| Smile\_Click() | Вызов функции «New\_Game()» при нажатии на смайлик |
| Beginner\_Click() | Вызов функции «Beginner\_Mode()» при нажатии на определённую кнопку |
| Enthusiast\_Click() | Вызов функции «Enthusiast\_Mode()» при нажатии на определённую кнопку |
| Professional\_Click() | Вызов функции «Professional\_Mode()» при нажатии на определённую кнопку |
| Window\_KeyDown() | Отслеживание нажатий определённых кнопок |
| Records\_Click() | Вызов функции «Transition\_Records\_Window()» при нажатии на определённую кнопку |
| Special\_Click() | Вызов функции «Transition\_Special\_Window()» при нажатии на определённую кнопку |
| Exit\_Click() | Выход из программы |
| RB\_100\_Checked() | Передача параметров функции «Window\_Size()» и «MineField\_Elements\_Size()» |
| RB\_150\_Checked() | То же |
| RB\_200\_Checked() | То же |

Таблица 4.2 – Модуль «Special\_Mode»

|  |  |
| --- | --- |
| Функция | Действие |
| NewGame\_Click() | Проверка нет ли ошибок заполнения, если нет начать новую игру |
| Only\_Number() | Разрешён ввод только цифр |
| Without\_a\_Space() | Запрещён ввод пробелов |
| Fill\_TBH() | Проверка нет ли ошибок в заполнении поля «Высота» и вывод определённого цвета при ошибке и её отсутствии |
| Fill\_TBW() | Проверка нет ли ошибок в заполнении поля «Ширина» и вывод определённого цвета при ошибке и её отсутствии |
| Fill\_TBM() | Проверка нет ли ошибок в заполнении поля «Мины» и вывод определённого цвета при ошибке и её отсутствии |

Таблица 4.3 – Модуль «Records\_Window»

|  |  |
| --- | --- |
| Функция | Действие |
| Records\_Window() | Отображение таблицы в соответствии с режимом и проверка победы и побития рекорда для записи результата в таблицу |
| CheckTableHit() | Запись имени и рекорда в DataGrid |
| UpdateToTable() | Перезапись БД новыми данными из DataGrid |
| selectALLData() | Подключение к БД и вывод содержимого определённой таблицы в DataGrid |
| Transition\_Nick\_Name\_Window() | Переход на форму «Ввод имени» |
| RB\_BeginnerMode\_Checked() | Передача данных «selectALLData()» при изменении Checkbox «RB\_BeginnerMode» |

Продолжение таблицы 4.3

|  |  |
| --- | --- |
| RB\_EnthusiastMode\_Checked() | Передача данных «selectALLData()» при изменении Checkbox «RB\_EnthusiastMode» |
| RB\_ProfessionalMode\_Checked() | Передача данных «selectALLData()» при изменении Checkbox «RB\_ProfessionalMode» |
| Input\_Name() | Вызов функции «Transition\_Nick\_Name\_Window()» и проверка имени пользователя на пустоту |

Таблица 4.4 – Модуль «Nick\_Name»

|  |  |
| --- | --- |
| Функция | Действие |
| Nick\_NameTB\_TextChanged() | Подсвечивать определённым цветом, если пользователь ввёл или не ввёл имя |
| Window\_Closing() | Подтверждение закрытия окна |
| Without\_a\_Space() | Запрещён ввод пробелов |
| Button\_Click() | Подтверждение ввода имени |

### Обращение к программе

Для запуска программы необходимо дважды кликнуть левой кнопкой мыши по файлу «S1WP.exe».

### Входные и выходные данные

Входные данные – данные вводимые или выбранные пользователем в ходе работы программы. К ним относится ввод имени в таблицу рекордов, а также ручная установка размеров игрового поля и количества мин на нём.

Выходные данные – данные, полученные в ходе работы пользователя с программой. К ним относится запись введённого пользователем имени в базу данных (БД), функции вывода игрового поля на экран монитора, количество оставшихся на поле мин и время на таймере, прошедшее с начала игры.

### Сообщения программисту

В данном курсовом проекте нет сообщений, предназначенных программисту.

## Руководство пользователя

### Назначение программы

Данная программа предназначена для обучения пользователя быстрому и точному использованию левой и правой кнопок мыши, а также развитие логического мышления посредством игры.

Применяя эту программу, пользователи имеют следующие возможности:

* игра на разных уровнях сложности;
* собственноручное создание игры с желаемыми параметрами;
* увеличение или уменьшение размеров интерфейса;
* просмотр существующих рекордов;
* внесение своего имени в таблицу рекордов;
* просмотр справки к игре.

### Условия выполнения программы

Данная программа будет выполняться при соблюдении всех условий, указанных в разделах 4.1.2 – 4.1.4.

### Выполнение программы

Для запуска программы нужно двойным щелчком левой кнопки мыши нажать на иконку «S1WP.exe». (рисунок 4.2)



Рисунок 4.2 – Приложение для запуска программы

Запустив программу, пользователь попадает на главную форму программы, где может выполнить большую часть её функций, выбор которых осуществляется при помощи компьютерной мыши или нажатием определённых клавиш на клавиатуре (рисунок 4.3).

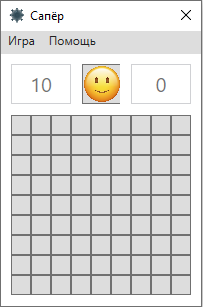


Рисунок 4.3 – Главное окно программы

Вовремя выполнения программы пользователю будут выводиться подсказки о том, при нажатии на какие клавиши будет вызвана та или иная функция, полный список клавиш и функций, за которые они отвечают можно посмотреть в «Приложение А».

В целях демонстрации всех возможностей программы пройдёмся по большинству из них перед началом самой игры. Большинство функций программы можно найти в меню, расположенном в верхней части окна нажав на пункты «Игра» или «Помощь» (рисунок 4.4).

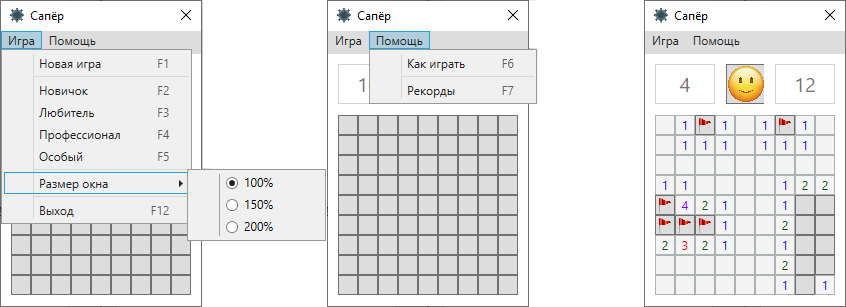


Рисунок 4.4 – Содержимое пунктов меню «Игра» и «Помощь»

Если вы запускаете программу в первый раз и понятия не имеете как ей пользоваться или давно не пользовались программой и вам требуется освежить память, вы можете ознакомится с инструкцией перейдя в пункт меню «Помощь» и выбрав подпункт «Как играть». Нажав на него левой кнопкой мыши или вызвав его при помощи клавиши «F6» перед вами откроется следующее окно (рисунок 4.5).

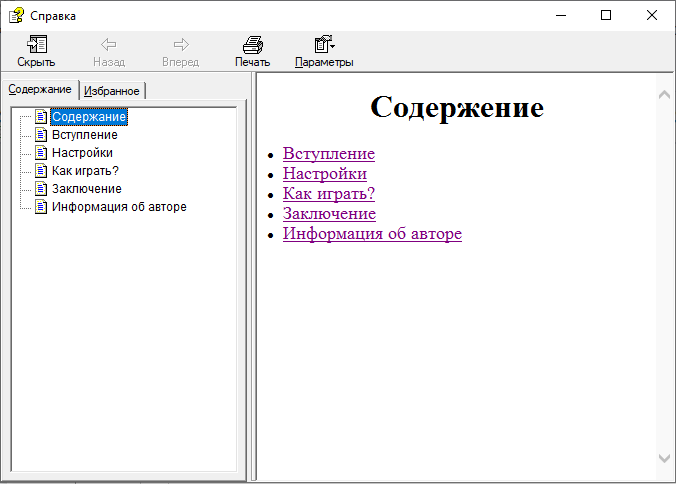


Рисунок 4.5 – Окно справки

Это окно справки, как я уже говорил ранее, оно нужно для того, чтобы не тратить времени на поиски информации о том, как пользоваться программой в интернете или суметь ей воспользоваться даже, если интернета нет.

Перемещаться между вкладками можно как при помощи содержания в левой части окна, так и при помощи открытого изначально в правой части, как на рисунке 4.5.

Всего в справке содержится 6 страниц:

* «Содержание»;
* «Вступление»;
* «Настройки»;
* «Как играть?»;
* «Заключение»;
* «Информация об авторе».

На первой странице, как уже многие поняли отображается содержание справки. Во вступлении рассказывается про игру «Сапёр» в целом (рисунок 4.6).

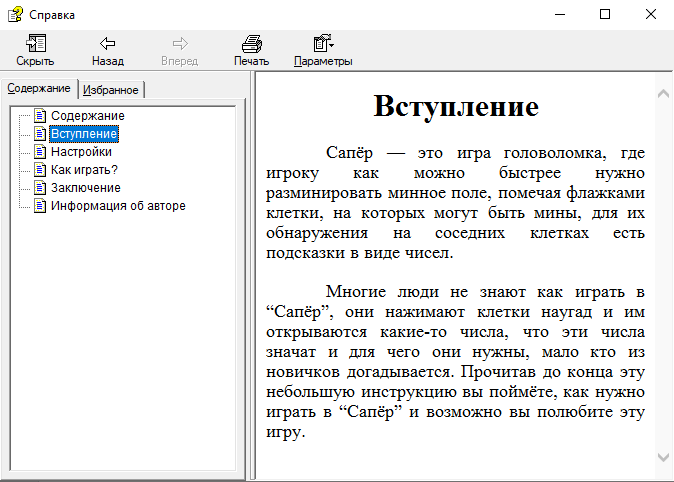


Рисунок 4.6 – Окно справки на странице «Вступление»

«Настройки» помогут разобраться как изменить уровень сложности, увеличить или уменьшить размер окна и здесь также будет выведен список «горячих клавиш» и их функций (рисунок 4.7).

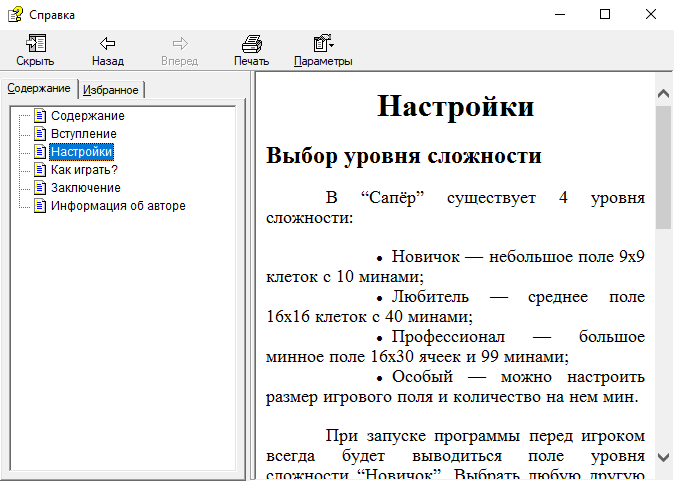


Рисунок 4.7 – Окно справки на странице «Настройки»

На следующей страничке будут рассказаны правила игры, как нужно и как не нужно играть чтобы победить и так далее (рисунок 4.8).

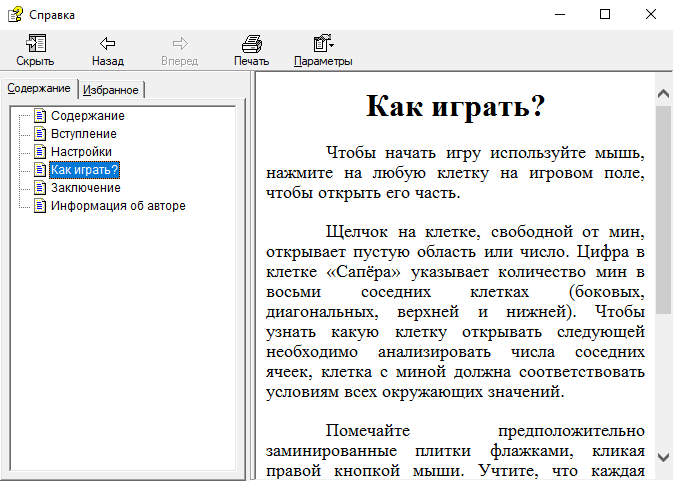


Рисунок 4.8 – Окно справки на странице «Как играть?»

В «Заключение» будут подведены итоги и даны советы по игре (рисунок 4.9).

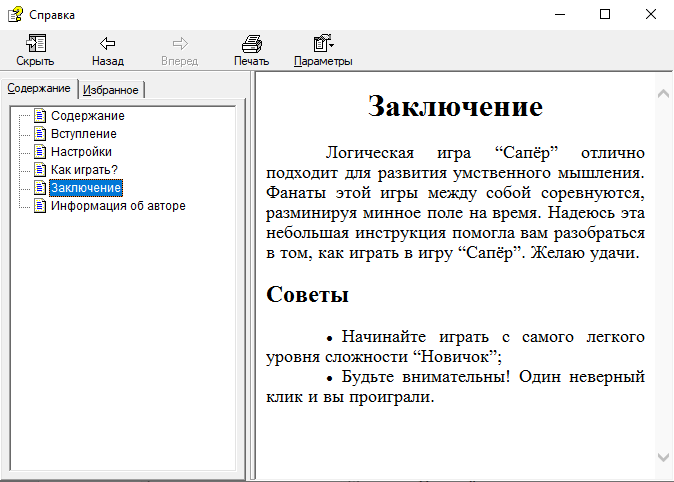


Рисунок 4.9 – Окно справки на странице «Заключение»

В «Информации об авторе» будут указаны контакты автора программы (рисунок 4.10).

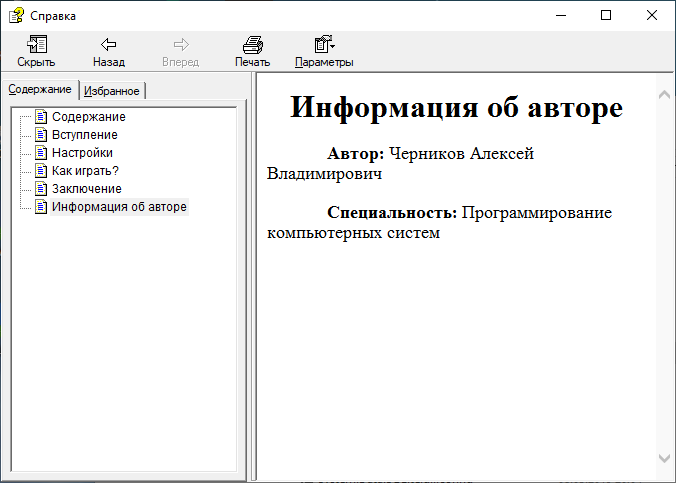


Рисунок 4.10 – Окно справки на странице «Информации об авторе»

Чтобы ознакомиться с рекордами других игроков выберите пункт меню «Помощь» и подпункт «Рекорды». Нажав на него левой кнопкой мыши или вызвав его при помощи клавиши «F7» перед вами откроется следующее окно (рисунок 4.11).

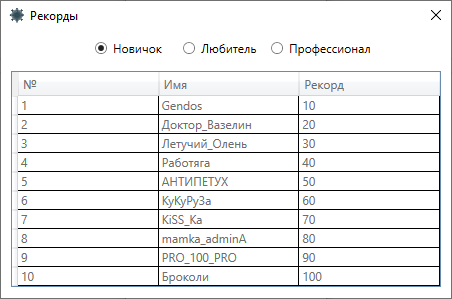


Рисунок 4.11 – Окно «Рекорды» с открытой таблицей рекордов уровня сложности «Новичок»

В этом окне вы можете просмотреть рекорды для трёх основных уровней сложности игры, а именно «Новичок», «Любитель», «Профессионал». Таблицы рекордов для уровня сложности «Особый» не была включена в список, так как в этом режиме игрок сам задаёт размер поля и количество мин, в связи с чем отслеживание результата не имеет смысла.

Посмотреть любую из доступных таблиц мы можем, нажимая на переключатели «Новичок», «Любитель» и «Профессионал» в верхней части окна (рисунок 4.12, 4.13).



Рисунок 4.12 – Окно «Рекорды» с открытой таблицей рекордов уровня сложности «Любитель»

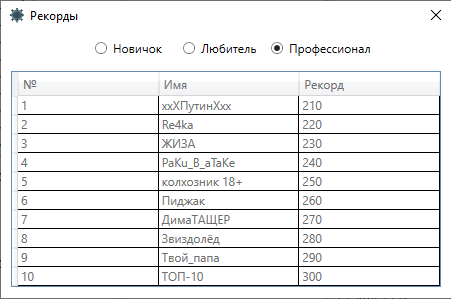


Рисунок 4.13 – Окно «Рекорды» с открытой таблицей рекордов уровня сложности «Профессионал»

Так как таблица рекордов защищена, ввести своё имя в её список можно только победив в игре за наименьшее время.

Вернёмся на главную форму и рассмотрим функции, доступные из пункта меню «Игра». Начнём с подпункта «Размер окна», он нужен для увеличения или уменьшения размеров интерфейса. Нажав на него перед вами выпадет список из трёх следующих пунктов:

«100%» – стандартный размер интерфейса;

«150%» – интерфейс, увеличенный на 50%;

«200%» – интерфейс, увеличенный на 100%.

Соответственно нажав любой из этих пунктов, в зависимости от текущего размера интерфейса вы либо увеличите, либо уменьшите размеры интерфейса. Также вы можете увеличивать, нажимая на клавишу «+» или уменьшать интерфейс, нажимая на клавиши «-» (клавиши «+» и «-» не работают на цифровой клавиатуре). На рисунках 4.14 – 4.16 продемонстрированы разные размеры интерфейса.

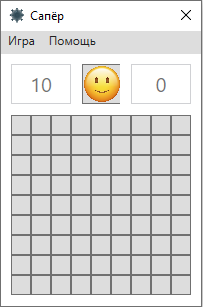


Рисунок 4.14 – Стандартный размер интерфейса («100%»)

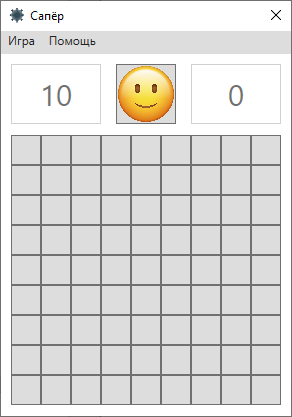


Рисунок 4.15 – Интерфейс, увеличенный на 50% («150%»)

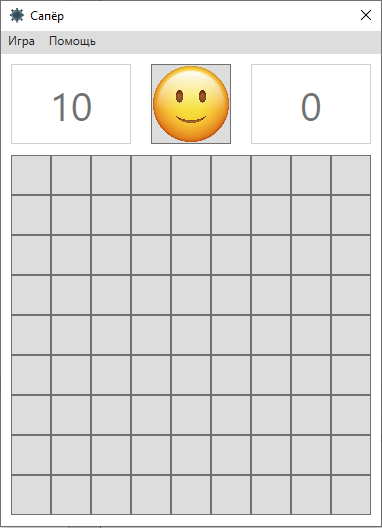


Рисунок 4.16 – Интерфейс, увеличенный на 100% («200%»)

Теперь рассмотрим все существующие уровни сложности, всего их 4:

* «Новичок» – небольшое поле 9х9 клеток с 10 минами (клавиша «F2»);
* «Любитель» – среднее поле 16х16 клеток с 40 минами (клавиша «F3»);
* «Профессионал» – большое минное поле 16х30 ячеек и 99 минами (клавиша «F4»);
* «Особый» – можно настроить размер игрового поля и количество на нем мин (клавиша «F5»).

Соответственно активировать их можно, выбрав интересующую вас сложность мышкой, либо определённой клавишей. Первые 3 уровня сложности продемонстрированы на рисунках 4.17 – 4.19.

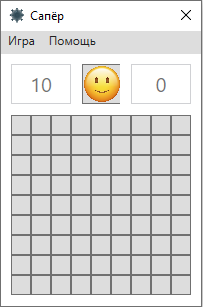


Рисунок 4.17 – Уровень сложности «Новичок»

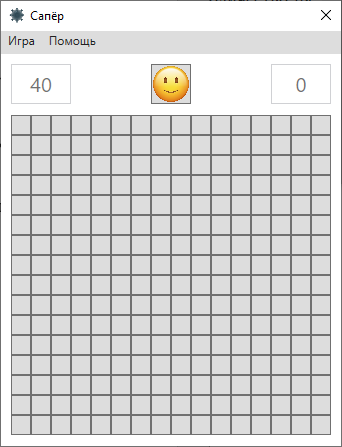


Рисунок 4.18 – Уровень сложности «Любитель»

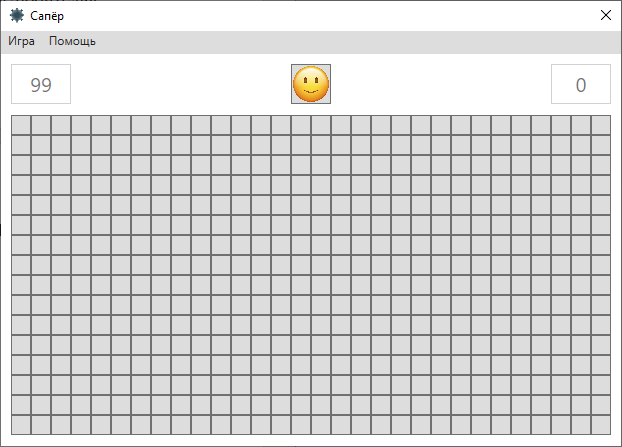


Рисунок 4.19 – Уровень сложности «Профессионал»

Подробнее остановимся на режиме «Особый». Выбрав этот режим перед вами откроется окно, указанное на рисунке 4.20.

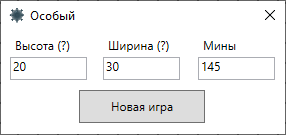


Рисунок 4.20 – Окно «Особый»

В этом окне задаются параметры создаваемой игры. По умолчанию предложено создать поле размерами 20х30, включающее 145 мин, если вы хотите задать другие размеры поля и количество мин просто выберите интересующий вас параметр и напишите своё значение. Чтобы заполнить поля правильно нужно:

* заполнять поля только целыми числами;
* заполнить все 3 поля;
* высота игрового поля не может быть равна 0 или превышать 20;
* ширина игрового поля не может быть меньше 9 или превышать 40.

Если поле заполнено неправильно оно будет подсвечено красным, как на рисунке 4.21.

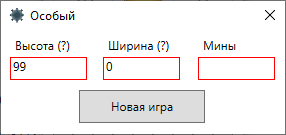


Рисунок 4.21 – Окно «Особый» с вводом неверных значений

Если вы вроде бы всё ввели правильно, но не понимаете почему поле до сих пор подсвечивается красным наведите мышку на знак «?» для получения подсказки (рисунок 4.22).

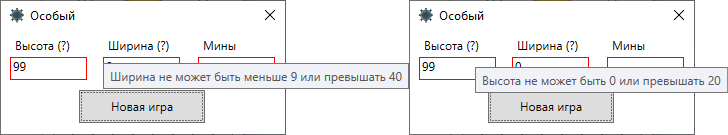


Рисунок 4.22 – Окно «Особый» с выводом подсказки

Если в заполнении полей не допущено ошибок окно будет выглядеть следующим образом (рисунок 4.23).

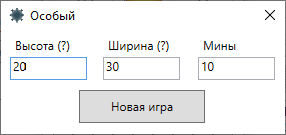


Рисунок 4.23 – Окно «Особый» с вводом верных значений

Теперь нажав на кнопку «Новая игра», вы попадёте на главное окно, на котором и будет отображено игровое поле с вашими настройками (рисунок 4.24).

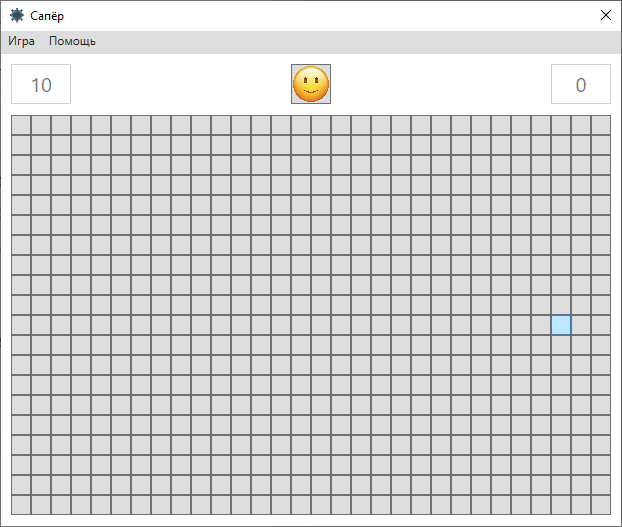


Рисунок 4.24 – Главное окно с игровым полем сложности «Особый»

Далее перейдём к самой игре (игровому процессу). Играть будем на уровне сложности «Новичок». Итак, начнём. Чтобы открыть ячейку на игровом поле необходимо щёлкнуть на неё левой кнопкой мыши. В зависимости от того, что хранится в ячейке может вывестись мина, в результате чего игра будет закончена (рисунок 4.25).

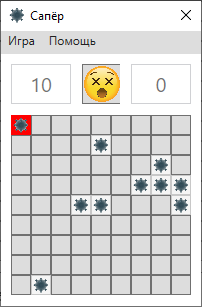


Рисунок 4.25 – При нажатии левой кнопкой мыши на мину (поражение)

Если это произошло не расстраивайтесь, нажмите на смайлик рядом с игровым полем, либо в пункте меню «Игра» выберите подпункт «Новая игра» (клавиша «F1»), чтобы начать новую игру (рисунок 4.26).

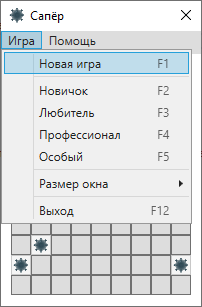


Рисунок 4.26 – Запуск новой игры

Или может вывестись число и открыться все пустые соседние ему клетки, в результате чего игра продолжится (рисунок 4.27).

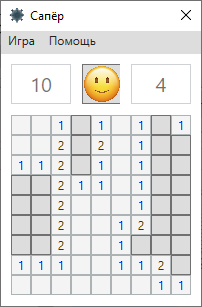


Рисунок 4.27 – При нажатии левой кнопкой мыши на пустую клетку

Если вы уверены, что в какой-либо ячейке находится мина вы, можете отметить её флажком, нажав на правую кнопку мыши (рисунок 4.28).



Рисунок 4.28 – Установка флажков на выбранные клетки

Если вы хотите убрать флажок с клетки просто щёлкните на него правой кнопкой мыши ещё раз.

Когда будут открыты все пустые клетки на поле игра закончится, это будет означать, что вы победили (рисунок 4.29).

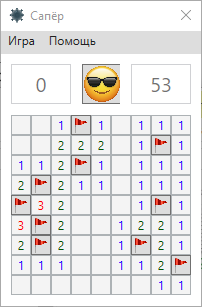


Рисунок 4.29 – Победа в игре

В зависимости от того, как быстро вы победили в игре вам может быть предложено внести своё имя в список рекордов (рисунок 4.30).

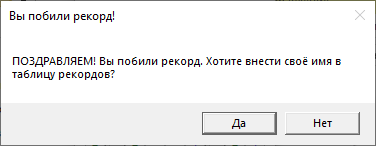


Рисунок 4.30 – Предложение внести своё имя в список рекордов

Если вы откажитесь, вас перенесёт на окно «Рекорды». Если вы согласитесь, перед вами откроется окно ввода имени (рисунок 4.31).

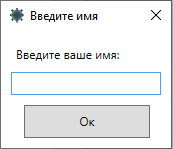


Рисунок 4.31 – Окно ввода имени

Здесь вы можете ввести своё имя, оно не может превышать более 22 символов и содержать пробелы. Если эти требования не будут соблюдены, программа укажет на ошибку (рисунок 4.32).

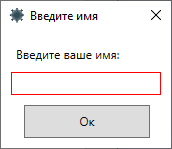


Рисунок 4.32 – Окно ввода имени, ошибка ввода

Если вы передумали вносить своё имя в список рекордов, тогда нажмите на крестик, для того, чтобы закрыть окно. Это действие необходимо подтвердить, ведь сделав его, ваш рекорд не будет записан (рисунок 4.33).

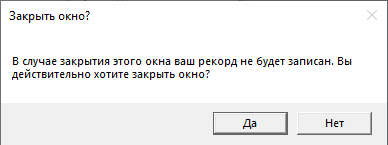


Рисунок 4.33 – Окно ввода имени, подтверждение закрытия окна

Нажав «Да», вы перейдёте к окну «Рекорды», нажав «Нет», вернётесь к окну ввода имени. Я же введу имя, для внесения его в таблицу рекордов (рисунок 4.34).

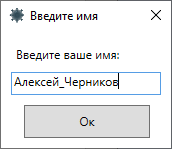


Рисунок 4.34 – Окно ввода имени, введение имени

Нажмём «Ок» и перед нами откроется окно «Рекорды», уже с записанным в таблицу именем, введённого нами и нашим рекордом (рисунок 4.35).

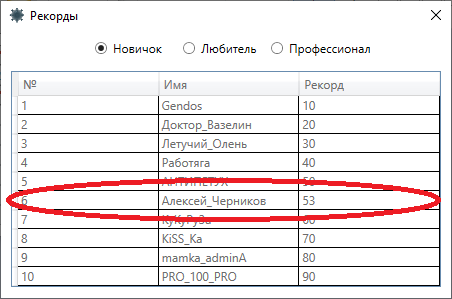


Рисунок 4.35 – Окно «Рекорды», с добавленным в его таблицу нашего рекорда

Начав новую игру и снова побив рекорд перед нами появится следующее сообщение с предложением использовать введённое нами ранее имя (рисунок 4.36).

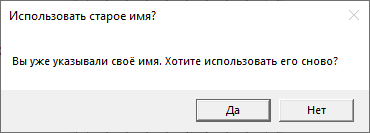


Рисунок 4.36 – Предложение использовать введённое раннее имя

Если вы откажетесь, то перейдёте к окну ввода имени, если согласитесь, перейдёте к окну «Рекорды» уже с записанным в таблицу именем, введённого нами ещё в первый раз и нашим новым рекордом (рисунок 4.37).

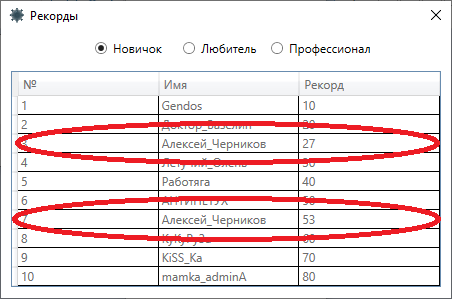


Рисунок 4.37 – Использование старого имени

Теперь расскажу про последнюю функцию программы, а именно «Выход». Помимо простого нажатия на крестик в правом верхнем углу окна для закрытия программы, можно также воспользоваться функцией, которая расположена в пункте меню «Игра». Нажав на неё левой кнопкой мыши или нажав клавишу «F12», программа завершит свою работу.

### Сообщения пользователю

Сообщения, предназначенные пользователю продемонстрированы на рисунках 4.38 – 4.43.

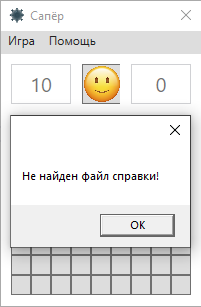


Рисунок 4.38 – Сообщение об отсутствии файла справки

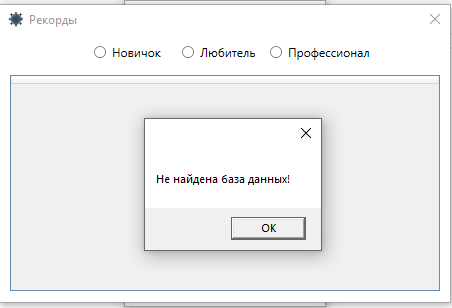


Рисунок 4.39 – Сообщение об отсутствии базы данных

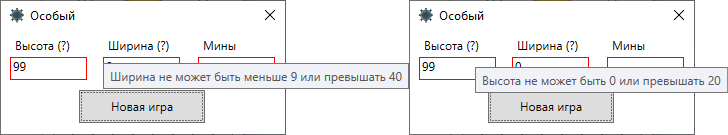


Рисунок 4.40 – Сообщения-подсказки при вводе неверных данных

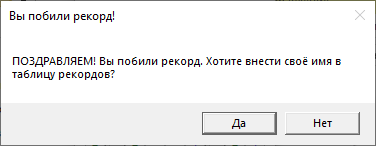


Рисунок 4.41 – Сообщение с предложением внести имя в список рекордов

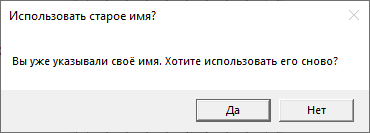


Рисунок 4.42 – Сообщение с предложением внести старое имя в список рекордов

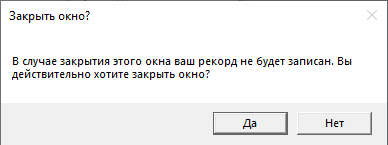


Рисунок 4.43 – Сообщение подтверждения выхода из окна для ввода имени

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте была создана игра «Сапёр». При написании проекта была изучена специальная литература, алгоритм работы аналогов игры, новые библиотеки и способы написания кода на языке С#.

Изложенное мною раннее подразумевает, что все цели проекта были выполнены, функции реализованы и протестированы.

Достоинствами проекта можно считать:

* Простота и удобство использования;
* Наличие разных уровней сложности;
* Наличие таблицы рекордов для трёх основных уровней сложности;
* Возможность установки флажков;
* Наличие таймера и счётчика оставшихся мин;
* Изменение эмоций смайлика в зависимости от результатов игры;
* Наличие «горячих» клавиш для каждой основной функции программы;
* Наличие справки с инструкцией и информацией об авторе.

В проект небыли добавлены:

* Кастомизация интерфейса.

Проведенные испытания показали корректность работы программы. Следовательно, все вышеизложенное позволяет рекомендовать эту игру для использования в целях развития логического мышления, обучению пользователя быстрому и точному использованию левой и правой кнопок мыши, и параллельного развлечения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Стандарты

1. ГОСТ 7.1. – 2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
2. ГОСТ 7.32 – 2001. Система стандартов по информацию, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.
3. ГОСТ 7.82 – 2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. –21 с.
4. Единая система программной документации. – М.: Стандартинформ, 2005. –128 с.

Интернет-ресурсы

1. Требования к оформлению пояснительной записки. – [Электронный ресурс]. – Режим доступа: https://pandia.ru/text/77/378/34277.php
2. Стандарты и шаблоны для ТЗ на разработку ПО. – [Электронный ресурс]. – Режим доступа: https://habr.com/post/328822/
3. Полное руководство по языку программирования С# 7.0 и платформе .NET 4.7. – [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/
4. Руководство по C#. События. – [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/csharp/charp\_theory/level10/10\_7.php
5. Как написать игру сапёр (минёр)? Алгоритм игры сапёр (минёр). – [Электронный ресурс]. – Режим доступа: http://orionxl.ru/morfologiya-igr-kak-napisat-ili-sozdat-igru/kak-napisat-igru-saper.html
6. SQLite + C#. – [Электронный ресурс]. – Режим доступа: https://devpractice.ru/sqlite-c/
7. Работа с URI. – [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/csharp/web/level2/2\_2.php
8. События ввода с клавиатуры. – [Электронный ресурс]. – Режим доступа: https://professorweb.ru/my/WPF/base\_WPF/level5/5\_8.php
9. Как создать CHM файл (HTM2CHM и HTML Help Workshop в действии). – [Электронный ресурс]. – Режим доступа: http://www.oszone.net/3789
10. Как оформить диплом по ГОСТу? Требования к оформлению дипломной работы по ГОСТу 2018. – [Электронный ресурс]. – Режим доступа: https://studlance.ru/blog/oformlenie-diplomnoy-po-gostu

# ПРИЛОЖЕНИЕ А

Список горячих клавиш программы

Таблица А.1 – Список горячих клавиш программы

|  |  |
| --- | --- |
| Клавиша | Действие |
| F1 | Новая игра |
| F2 | Уровень сложности «Новичок» |
| F3 | Уровень сложности «Любитель» |
| F4 | Уровень сложности «Профессионал» |
| F5 | Уровень сложности «Особый» |
| F6 | Открыть справку |
| F7 | Открыть окно рекордов |
| F12 | Выход из программы |
| + | Увеличение размеров интерфейса |
| - | Уменьшение размеров интерфейса |

# ПРИЛОЖЕНИЕ Б

Листинг программы

Модуль «MainWindow»

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;

using System.Collections;

using System.Windows.Threading;

namespace S1WP

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public enum TypeButton

{

Mine,

Flag,

Number,

Emotion

}

public partial class MainWindow : Window

{

#region Переменные

public static int x = 9, y = 9, CountMine = 10, ModeTag = 1, timesec = 0;

static int CountFlag = CountMine, ActualStep = 0;

public static bool WinOrLose = false;

static bool result = false;

int[,] field = new int[x, y];

int HSize = 20, WSize = 20, FSize = 12, SBSize = 40, TBWSize = 60, TBFSize = 20, GSS = 0;

Button[] ButtonList = new Button[x \* y];

BitArray GameArray= new BitArray(x \* y);

Random rnd = new Random();

DispatcherTimer timer = null;

Uri uriMine = new Uri("pack://application:,,/Mine.png");

Uri uriFlag = new Uri("pack://application:,,/Flag.png");

Uri uriEmotion1 = new Uri("pack://application:,,/Emotion1.png");

Uri uriEmotion2 = new Uri("pack://application:,,/Emotion2.png");

Uri uriEmotion3 = new Uri("pack://application:,,/Emotion3.png");

#endregion

public MainWindow()

{

InitializeComponent();

Installation\_Mine();

RB\_100.IsChecked = true;

}

#region Игра

void Installation\_Mine(bool breaks = false, int count = 0)

{

SetButton(Smile, TypeButton.Emotion);

timer = new DispatcherTimer();

timesec = 0;

timer.Tick += new EventHandler(timer\_tick);

timer.Interval = new TimeSpan(0, 0 ,0 , 0, 1000);

TimerTB.Text = Convert.ToString(timesec);

result = false;

CountFlag = CountMine;

ActualStep = 0;

CountFlagTB.Text = Convert.ToString(CountFlag);

Minefield.Width = HSize \* x;

Minefield.Height = WSize \* y;

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

{

if (count == CountMine)

break;

if (!GameArray[i])

GameArray[i] = rnd.Next(0, 1000) < 50 && rnd.Next(0, 1000) > 800 ? true : false;

else

continue;

if (GameArray[i])

count++;

}

if (count < CountMine)

Installation\_Mine(true, count);

if (breaks)

return;

Window\_Size();

Creation\_Minefield(x \* y);

}

private void Creation\_Minefield(int count)

{

Minefield.Children.Clear();

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

{

Button t = new Button()

{

FontSize = FSize,

Width = WSize,

Height = HSize,

Tag = i

};

t.Click += new RoutedEventHandler(button\_Click);

t.MouseRightButtonUp += new MouseButtonEventHandler(rightbutton\_Click);

Minefield.Children.Add(t);

ButtonList[i] = t;

}

}

void rightbutton\_Click(object sender, MouseButtonEventArgs e)

{

if (result == false)

{

timer.Start();

Button button = (Button)sender;

int index = (int)button.Tag;

if (button.Content == null && CountFlag > 0)

{

SetButton(button, TypeButton.Flag);

CountFlag--;

CountFlagTB.Text = Convert.ToString(CountFlag);

}

else if (button.Content!=null)

{

button.Content = null;

CountFlag++;

CountFlagTB.Text = Convert.ToString(CountFlag);

}

}

}

void button\_Click(object sender, RoutedEventArgs e)

{

if (result == false)

{

Button button = (Button)sender;

int index = (int)button.Tag;

if (GameArray[index]) //Lose

{

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

if (GameArray[i])

SetButton(ButtonList[i], TypeButton.Mine);

SetButton(ButtonList[index], TypeButton.Mine, 1);

result = true;

timer.Stop();

SetButton(Smile, TypeButton.Emotion, 2);

}

else

{

timer.Start();

Step(index);

if (ActualStep == x \* y - CountMine) //Win

{

if (ModeTag != 0)

WinOrLose = true;

timer.Stop();

result = true;

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

if (GameArray[i])

SetButton(ButtonList[i], TypeButton.Flag);

CountFlag = 0;

CountFlagTB.Text = Convert.ToString(CountFlag);

SetButton(Smile, TypeButton.Emotion, 1);

if (ModeTag != 0)

{

Transition\_Records\_Window();

}

}

}

}

}

void SetButton(Button button, TypeButton type, int number = 0)

{

BitmapImage bitmap = new BitmapImage();

if (type == TypeButton.Emotion)

{

switch (number)

{

case 0: bitmap = new BitmapImage(uriEmotion1); break;

case 1: bitmap = new BitmapImage(uriEmotion2); break;

case 2: bitmap = new BitmapImage(uriEmotion3); break;

}

}

if (type == TypeButton.Mine)

{

bitmap = new BitmapImage(uriMine);

if (number == 1)

{

button.Background = new SolidColorBrush(Color.FromRgb(255, 0, 0));

button.BorderBrush = new SolidColorBrush(Color.FromRgb(255, 0, 0));

}

else

{

button.Background = new SolidColorBrush(Color.FromRgb(245, 245, 245));

button.BorderBrush = new SolidColorBrush(Color.FromRgb(200, 200, 200));

}

}

if (type == TypeButton.Flag)

bitmap = new BitmapImage(uriFlag);

if (type != TypeButton.Number)

{

Image img = new Image()

{

Source = bitmap

};

Grid grid = new Grid();

grid.Children.Add(img);

button.Content = grid;

}

if (type == TypeButton.Number)

{

if (button.Content != null)

{

CountFlag++;

CountFlagTB.Text = Convert.ToString(CountFlag);

}

if (number == 0)

{

button.Content = null;

}

else

button.Content = number;

button.IsEnabled = false;

}

}

void Step(int index)

{

int[] t = new int[8];

int row = index / x;

int column = row \* x - index;

t[0] = x \* (row - 1) + (index % x);

t[1] = x \* (row + 1) + (index % x);

t[2] = t[0] + 1;

t[3] = t[0] - 1;

t[4] = t[1] + 1;

t[5] = t[1] - 1;

t[6] = index + 1;

t[7] = index - 1;

int count = 0;

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

if (ContainsIndex(t[i], index, row, column))

if (GameArray[t[i]])

count++;

if (count == 0)

{

ButtonList[index].IsEnabled = false;

SetButton(ButtonList[index], TypeButton.Number);

ActualStep += 1;

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

if (ContainsIndex(t[i], index, row, column))

if (ButtonList[t[i]].IsEnabled)

Step(t[i]);

}

else

{

SetButton(ButtonList[index], TypeButton.Number, count);

switch (count)

{

case 1: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(0, 0, 255)); break;

case 2: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(0, 82, 0)); break;

case 3: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(255, 0, 0)); break;

case 4: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(139, 0, 255)); break;

case 5: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(139, 0, 0)); break;

case 6: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(8, 232, 222)); break;

case 7: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(0, 5, 71)); break;

case 8: ButtonList[index].Foreground = new SolidColorBrush(Color.FromRgb(87, 87, 87)); break;

}

ActualStep += 1;

}

}

bool ContainsIndex(int index, int baseInd, int baseRow, int baseCol)

{

if (index >= 0 && index < GameArray.Length)

{

int row = index / x;

int column = row \* x - index;

if (Math.Abs(baseRow - row) > 1)

return false;

if (Math.Abs(baseCol - column) > 1)

return false;

return true;

}

else

return false;

}

private void timer\_tick(object sender, EventArgs e)

{

timesec++;

TimerTB.Text = Convert.ToString(timesec);

}

public void New\_Game()

{

WinOrLose = false;

timer.Stop();

GameArray = null;

GameArray = new BitArray(x \* y);

ButtonList = null;

ButtonList = new Button[x \* y];

Installation\_Mine();

}

void Window\_Size()

{

this.Width = (Minefield.Width = WSize \* x) + 36;

this.Height = (Minefield.Height = HSize \* y) + 94 + SBSize;

Main\_Menu.Width = this.Width;

Smile.Height = Smile.Width = SBSize;

TimerTB.Width = CountFlagTB.Width = TBWSize;

TimerTB.Height = CountFlagTB.Height = SBSize;

TimerTB.FontSize = CountFlagTB.FontSize = TBFSize;

Thickness positionDown = new Thickness(10, (44+SBSize), 0, 0);

Thickness positionRight = new Thickness((this.Width - TBWSize - 26), 33, 0, 0);

Minefield.Margin = positionDown;

TimerTB.Margin = positionRight;

}

private void Beginner\_Mode()

{

ModeTag = 1;

x = 9;

y = 9;

CountMine = 10;

New\_Game();

}

private void Enthusiast\_Mode()

{

ModeTag = 2;

x = 16;

y = 16;

CountMine = 40;

New\_Game();

}

private void Professional\_Mode()

{

ModeTag = 3;

x = 30;

y = 16;

CountMine = 99;

New\_Game();

}

private void Transition\_Records\_Window()

{

Records\_Window taskWindow = new Records\_Window();

taskWindow.Owner = this;

taskWindow.ShowDialog();

}

private void Transition\_Special\_Window()

{

ModeTag = 0;

Special\_Mode taskWindow = new Special\_Mode();

taskWindow.Owner = this;

taskWindow.ShowDialog();

}

private void MineField\_Elements\_Size()

{

Minefield.Width = HSize \* x;

Minefield.Height = WSize \* y;

for (int i = 0; i < (x \* y); i++)

{

ButtonList[i].Width = WSize;

ButtonList[i].Height = HSize;

ButtonList[i].FontSize = FSize;

}

}

private void Game\_Size\_Switch(bool D\_GSS)

{

if (D\_GSS == false && GSS > 0)

GSS--;

else if (D\_GSS == true && GSS < 2)

GSS++;

switch (GSS)

{

case 0: RB\_100.IsChecked = true; break;

case 1: RB\_150.IsChecked = true; break;

case 2: RB\_200.IsChecked = true; break;

}

}

private void Help()

{

try

{

string commandText = "Help.chm";

var proc = new System.Diagnostics.Process();

proc.StartInfo.FileName = commandText;

proc.StartInfo.UseShellExecute = true;

proc.Start();

}

catch

{

MessageBox.Show("Не найден файл справки!");

}

}

#endregion

#region Элементы формы

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

{

New\_Game();

}

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

{

Help();

}

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

{

New\_Game();

}

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

{

Beginner\_Mode();

}

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

{

switch (e.Key)

{

case Key.F1: New\_Game(); break; //Новая игра

case Key.F2: Beginner\_Mode(); break; //Новичок

case Key.F3: Enthusiast\_Mode(); break; //Любитель

case Key.F4: Professional\_Mode(); break; //Профессионал

case Key.F5: Transition\_Special\_Window(); break; //Особый

case Key.F6: Help(); break; //Как играть

case Key.F7: Transition\_Records\_Window(); break; //Рекорды

case Key.F12: this.Close(); break; //Выход

case Key.OemMinus: Game\_Size\_Switch(false); break; //Уменьшение размера игры

case Key.OemPlus: Game\_Size\_Switch(true); break; //Увеличение размера игры

}

}

private void RB\_100\_Checked(object sender, RoutedEventArgs e)

{

HSize = WSize = 20;

FSize = 12;

SBSize = 40;

TBWSize = 60;

TBFSize = 20;

MineField\_Elements\_Size();

Window\_Size();

}

private void RB\_150\_Checked(object sender, RoutedEventArgs e)

{

HSize = WSize = 30;

FSize = 15;

SBSize = 60;

TBWSize = 90;

TBFSize = 30;

MineField\_Elements\_Size();

Window\_Size();

}

private void RB\_200\_Checked(object sender, RoutedEventArgs e)

{

HSize = WSize = 40;

FSize = 20;

SBSize = 80;

TBWSize = 120;

TBFSize = 40;

MineField\_Elements\_Size();

Window\_Size();

}

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

{

Enthusiast\_Mode();

}

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

{

Transition\_Records\_Window();

}

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

{

Professional\_Mode();

}

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

{

Transition\_Special\_Window();

}

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

{

this.Close();

}

#endregion

}

}

Модуль «Special\_Mode»

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;

namespace S1WP

{

/// <summary>

/// Логика взаимодействия для Special\_Mode.xaml

/// </summary>

public partial class Special\_Mode : Window

{

public Special\_Mode()

{

InitializeComponent();

}

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

{

int h, w;

if (Special\_Height.Text == "" || Special\_Weight.Text == "" || Special\_Mine.Text == ""

|| Special\_Height.Text == "0" || Special\_Weight.Text == "0" || (h = Convert.ToInt32(Special\_Height.Text)) > 20

|| (w = Convert.ToInt32(Special\_Weight.Text)) < 9 || (w = Convert.ToInt32(Special\_Weight.Text)) > 40)

{

//Ничего

}

else

{

MainWindow.y = Convert.ToInt32(Special\_Height.Text);

MainWindow.x = Convert.ToInt32(Special\_Weight.Text);

MainWindow.CountMine = Convert.ToInt32(Special\_Mine.Text);

if (MainWindow.x < 9)

MainWindow.x = 9;

if (MainWindow.CountMine >= (MainWindow.x \* MainWindow.y))

MainWindow.CountMine = (MainWindow.x \* MainWindow.y) - 1;

(this.Owner as MainWindow).New\_Game();

this.Close();

}

}

void Only\_Number(object sender, TextCompositionEventArgs e)

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

private void Without\_a\_Space(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

private void Fill\_TBH(object sender, TextChangedEventArgs e)

{

int h = 0;

if (Special\_Height.Text == "" || (h = Convert.ToInt32(Special\_Height.Text)) > 20)

{

Special\_Height.BorderBrush = new SolidColorBrush(Color.FromRgb(255, 0, 0));

}

else if (Special\_Height.Text == "0")

{

Special\_Height.BorderBrush = new SolidColorBrush(Color.FromRgb(255, 0, 0));

}

else

Special\_Height.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

private void Fill\_TBW(object sender, TextChangedEventArgs e)

{

int w = 0;

if (Special\_Weight.Text == "" || (w = Convert.ToInt32(Special\_Weight.Text)) < 9

|| (w = Convert.ToInt32(Special\_Weight.Text)) > 40)

{

Special\_Weight.BorderBrush = new SolidColorBrush(Color.FromRgb(255, 0, 0));

}

else if (Special\_Weight.Text == "0")

{

Special\_Weight.BorderBrush = new SolidColorBrush(Color.FromRgb(255, 0, 0));

}

else

Special\_Weight.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

private void Fill\_TBM(object sender, TextChangedEventArgs e)

{

if (Special\_Mine.Text == "")

{

Special\_Mine.BorderBrush = new SolidColorBrush(Color.FromRgb(255, 0, 0));

}

else

Special\_Mine.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

}

}

Модуль «Records\_Window»

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.Data.SQLite;

using System.Data;

using Microsoft.Win32;

namespace S1WP

{

/// <summary>

/// Логика взаимодействия для Records\_Window.xaml

/// </summary>

public partial class Records\_Window : Window

{

public string FullPath;

public string RecGameMode;

public static DataTable data = new DataTable();

public int[] Record\_Array = new int[10];

int t = 0;

public Records\_Window()

{

InitializeComponent();

Table\_Records.IsEnabled=false;

switch (MainWindow.ModeTag)

{

case 0: RB\_BeginnerMode.IsChecked = true; break;

case 1: RB\_BeginnerMode.IsChecked = true; break;

case 2: RB\_EnthusiastMode.IsChecked = true; break;

case 3: RB\_ProfessionalMode.IsChecked = true; break;

}

if (MainWindow.WinOrLose == true)

{

for (int i = 0; i < 10; i++)

{

if (MainWindow.timesec <= Record\_Array[i])

{

t = i;

if (MessageBox.Show("ПОЗДРАВЛЯЕМ! Вы побили рекорд. Хотите внести своё имя в таблицу рекордов?", "Вы побили рекорд!", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

if (Nick\_Name.NNNickName == "" || Nick\_Name.NNNickName == null)

{

Input\_Name();

}

else

{

if (MessageBox.Show("Вы уже указывали своё имя. Хотите использовать его сново?", "Использовать старое имя?", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

CheckTableHit();

else

{

Input\_Name();

}

}

}

else

{

MainWindow.WinOrLose = false;

}

break;

}

}

}

}

private void CheckTableHit()

{

for (int i = 10; i > t; i--)

{

try

{

data.Rows[i][1] = Convert.ToString(data.Rows[i - 1][1]);

data.Rows[i][2] = Convert.ToString(data.Rows[i - 1][2]);

}

catch { }

}

data.Rows[t][1]=Convert.ToString(Nick\_Name.NNNickName);

data.Rows[t][2] = Convert.ToString(MainWindow.timesec);

MainWindow.WinOrLose = false;

UpdateToTable();

}

public void UpdateToTable()

{

SQLiteConnection connection = new SQLiteConnection(FullPath);

connection.Open();

for (int i = 0; i < 10; i++)

{

SQLiteParameter[] sQLiteParameter = new SQLiteParameter[2];

sQLiteParameter[0] = new SQLiteParameter("@Name", data.Rows[i][1].ToString());

sQLiteParameter[1] = new SQLiteParameter("@Record", data.Rows[i][2].ToString());

string sql = "Update " + RecGameMode + " set Name = @Name, Record = @Record where ID = '" + (i+1) + "';";

SQLiteCommand command = new SQLiteCommand(sql, connection);

command.Parameters.AddRange(sQLiteParameter);

SQLiteDataReader reader = command.ExecuteReader();

}

}

public void selectALLData()

{

try

{

data = null;

FullPath = "Data Source = Records.db;";

SQLiteConnection connection = new SQLiteConnection(FullPath);

connection.Open();

string sql = "select \* from " + RecGameMode + ";";

SQLiteCommand command = new SQLiteCommand(sql, connection);

SQLiteDataAdapter dataAdapter = new SQLiteDataAdapter(command);

data = new DataTable(RecGameMode);

dataAdapter.Fill(data);

data.Columns["ID"].ColumnName = "№";

data.Columns["Name"].ColumnName = "Имя";

data.Columns["Record"].ColumnName = "Рекорд";

Table\_Records.ItemsSource = data.DefaultView;

for (int i = 0; i < 10; i++)

{

Record\_Array[i] = Convert.ToInt32(data.Rows[i][2]);

}

}

catch

{

MessageBox.Show("Не найдена база данных!");

}

}

void Input\_Name()

{

Transition\_Nick\_Name\_Window();

if (Nick\_Name.NNNickName != "")

CheckTableHit();

}

public void Transition\_Nick\_Name\_Window()

{

Nick\_Name taskWindow = new Nick\_Name();

taskWindow.ShowDialog();

}

private void RB\_BeginnerMode\_Checked(object sender, RoutedEventArgs e)

{

RecGameMode = "Beginner";

selectALLData();

}

private void RB\_EnthusiastMode\_Checked(object sender, RoutedEventArgs e)

{

RecGameMode = "Enthusiast";

selectALLData();

}

private void RB\_ProfessionalMode\_Checked(object sender, RoutedEventArgs e)

{

RecGameMode = "Professional";

selectALLData();

}

}

}

Модуль «MainWindow»

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;

namespace S1WP

{

/// <summary>

/// Логика взаимодействия для Nick\_Name.xaml

/// </summary>

public partial class Nick\_Name : Window

{

public static string NNNickName;

public Nick\_Name()

{

InitializeComponent();

NNNickName = "";

Nick\_NameTB.Focus();

}

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

{

if (Nick\_NameTB.Text != "")

{

NNNickName = Convert.ToString(Nick\_NameTB.Text);

this.Close();

}

}

private void Without\_a\_Space(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

if (NNNickName == "")

{

if (MessageBox.Show("В случае закрытия этого окна ваш рекорд не будет записан. Вы действительно хотите закрыть окно?", "Закрыть окно?", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

MainWindow.WinOrLose = false;

}

else

{

e.Cancel = true;

}

}

}

private void Nick\_NameTB\_TextChanged(object sender, TextChangedEventArgs e)

{

if (Nick\_NameTB.Text == "")

{

Nick\_NameTB.BorderBrush = new SolidColorBrush(Color.FromRgb(255, 0, 0));

}

else

Nick\_NameTB.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

}

}

# ПРИЛОЖЕНИЕ В

Титульный лист презентации к защите курсового проекта

