Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по курсовому проекту
на тему

ПРОГРАММНОЕ СРЕДСТВО ИГРА «MineSweeper»

БГУИР КР 1-40 01 01 613 ПЗ

Выполнил:
студент гр. 851006 Крот Е. Д.

Руководитель: Жиденко А.Л.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

Лапицкая Н.В.

_____

(подпись)

_____2020 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Кроту Евгению Дмитриевичу*

1. Тема работы *Программное средство игра «MineSweeper»*

2. Срок сдачи законченной работы *08.12.2020г.*

3. Исходные данные к работе *Среда программирования MS Visual Studio 2019. Документация по Win32 API.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение
1 Анализ предметной области
2 Постановка задачи
3 Разработка программного средства
4 Тестирование и проверка работоспособности программного средства
5 Руководство пользователя программного средства
Заключение
Список использованных источников
Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)
Схема алгоритма в формате А1

6. Консультант по курсовой работе *Жиденко А.Л.*

7. Дата выдачи задания *01.10.2020.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):
Раздел 1. Введение к 20.10.2020г. – 10 % готовности работы;
Раздел 2 к 28.10.2020г. – 30% готовности работы
Раздел 3 к 15.11.2020г. – 60% готовности работы
Раздел 4 к 23.11.2020г. – 80% готовности работы
Раздел 5. Заключение. Приложения к 01.12.2020г. – 90% готовности работы;
Оформление пояснительной записки и графического материала к 05.12.2020г. – 100% готовности работы.
Защита курсового проекта с 01.12.2020г. по 10.12.2020г.

РУКОВОДИТЕЛЬ _____ *Жиденко А.Л.*
(подпись)

Задание принял к исполнению _____ *Крот Е.Д. 08.12.2020г.*
(дата и подпись студента)

СОДЕРЖАНИЕ

Введение

IT-специалист – широкое понятие, объединяющее в себе представителей многих профессий, работающих в области информационных технологий. Это программисты, разработчики, администраторы сетей и баз данных, модераторы, специалисты по робототехнике, по информационной безопасности, web-дизайнеры и 3D-художники. При этом, с проникновением информационных технологий во всё новые сферы деятельности, появляются новые профессии для IT-специалистов. По данным на сегодняшний день и мнению многих аналитиков [1] специалисты данной области являются востребованными и будут востребованы в ближайшем будущем. В частности, сегодня в мире появляется всё больше новых перспективных игровых разработок и проектов.

GameDev – сокращение от Game Development (разработка игр). В процесс разработки игрового приложения входят: разработка дизайна игрового процесса (геймплея), программирование игрового «движка» или использование готовых решений, разработка визуального концепта и его составляющих, создание графики и моделирование объектов, музыкального и звукового сопровождения, решение множества возникающих в процессе задач из различных сфер.

Игровая индустрия продолжает развиваться и с каждым годом её прибыль на рынке увеличивается. Таким образом, идея отработки навыков разработки игровых проектов является перспективным и интересным направлением.

Целью работы является разработка игры «MineSweeper» в жанре Головоломка. На данные момент существуют и разрабатываются множество игр данного жанра. По причине очень простой разработки со стороны программиста, а также большой популярности, со стороны пользователей жанр стабильно востребован и считается “классикой” среди игр. По причине крайней простоты геймплея, для любого приложения найдутся случайные и не очень аналоги в огромном количестве, работающие на разных платформах.

В рамках курсового проекта ставятся следующие задачи:

1. Осуществить постановку игровой задачи;
2. Выполнить обзор существующих аналогов, программных средств разработки компьютерной игры;
3. Спроектировать приложение;
4. Составить документацию на полученное приложение.

1 Анализ предметной области

1.1 Анализ существующих аналогов

Игра сапёр является одной из наиболее известных «Классических» игр операционной системы Windows. История игры «Сапёр» уходит в далекие 50-е года 20-го столетия. В то время это была, конечно же, не компьютерная игра, т.к. персональные компьютеры появились гораздо позже, а игра в большой картонной коробке. Версии данной игры так же распространялись в наборах игр для различных сборок Linux, существуют для MacOS, мобильных устройств и в виде браузерных приложений. В основном отличие всех вышеуказанных версий заключается в графическом оформлении, также некоторые версии имеют дополнительный функционал, зачастую слабо связанный с самим игровым процессом. Рассмотрим несколько наиболее популярных версий:

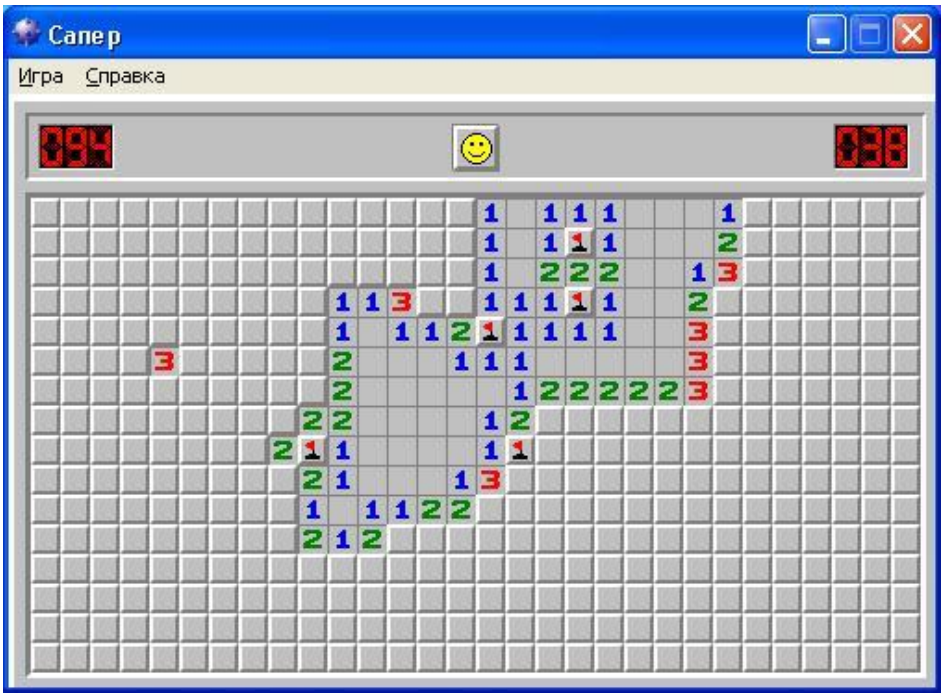

Рисунок 1.1 – «Сапёр» Windows XP

Одним из наиболее известных является «Сапёр», написанный для операционной системы Windows XP. Доступность игры во времена «до интернета», низкий порог вхождения, простота правил и одновременно интерес к решению привели к высокой популярности игры.

Ещё одной не менее популярной версией является дальнейшее развитие «Сапёра», но уже для Windows 7. Не только перенос, но и модификация программного средства спустя несколько лет свидетельствуют о популярности игры среди пользователей операционной системы. Самыми значимыми отличиями стали заметное изменение игрового интерфейса и появление анимации при поражении (взрыв мины, последовательный взрыв всех мин на игровом поле). Однако в целом игра не претерпела значительных изменений.

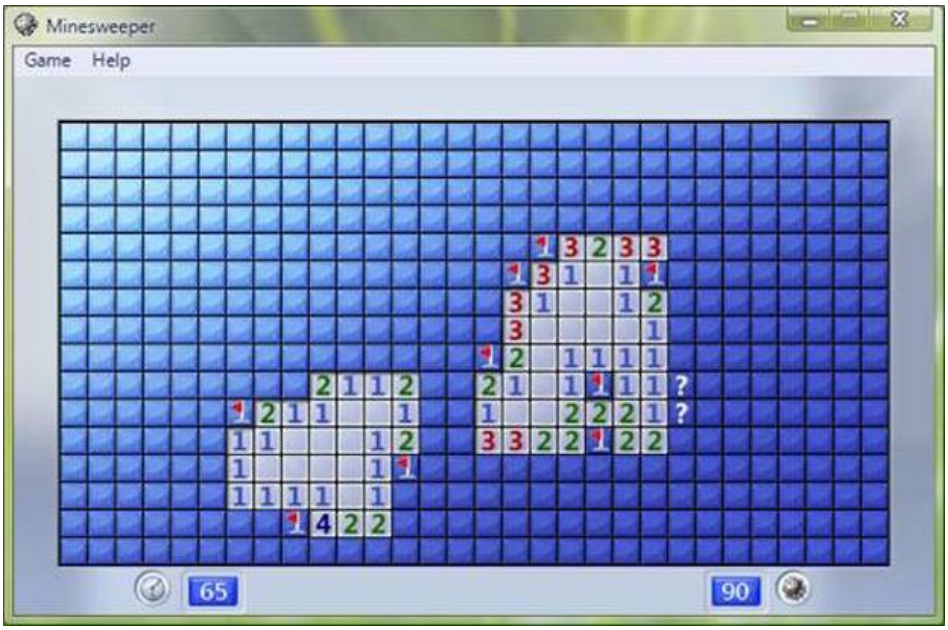

Рисунок 1.2 – «Сапёр» Windows 7

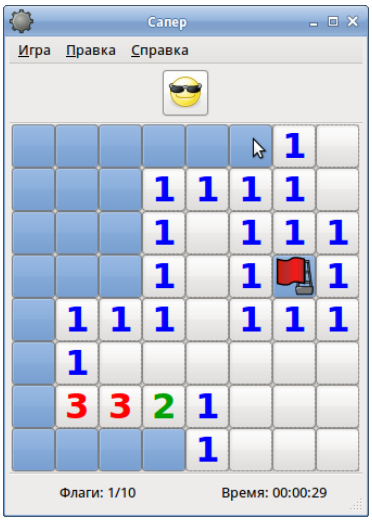

Рисунок 1.2 – «Сапёр» Gnome Games

Одной из наиболее популярных среди «Линуксовых» версий сапёра является версия, изданная Gnome Games. Первая версия игры появилась более двадцати лет назад, и до сих пор продолжает обновляться, что также свидетельствует о популярности игры. В отличие от версий ОС Windows, версия Gnome всегда имела самобытный дизайн, однако функционал так же не отличался от прообраза.

Интересными для рассмотрения могут быть дальнейшие развития игры «Сапёр», в значительной мере изменившие игровой процесс. Рассмотрим два варианта игры, Xbomb и emMines.

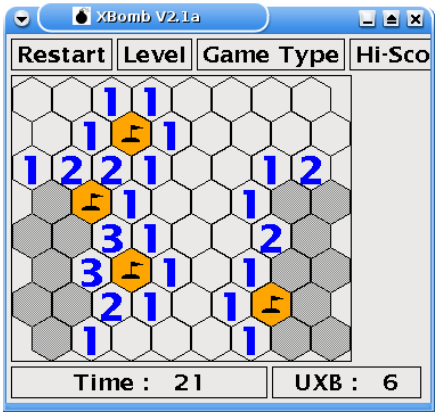

Рисунок 1.4 – «Сапёр» Xbomb

Xbomb представляет собой классический «Сапёр», однако перенесённый на поле с ячейками шестиугольной формы. Правила игры от этого, однако, не изменились, а дополнительного функционала в данную версию добавлено не было. На первый взгляд, такая игра может показаться сложнее, однако на самом деле с уменьшением количества соседних ячеек (с 8 до 6) немного снизилась и сложность. В целом, данный вариант является хорошим примером разнообразия «Сапёра». С точки зрения разработки же Xbomb является незначительно более сложной версией, однако отличие от обычного сапёра будет заключаться лишь в незначительном изменении структуры хранения информации о ячейках, а также алгоритмов подсчёта значения ячейки (количество мин в смежных ячейках), открытия ячейки и разбрасывания мин.

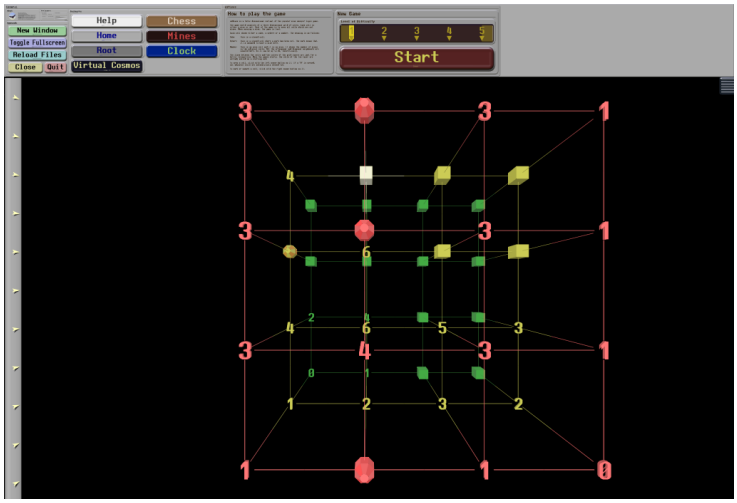

Рисунок 1.5 – «Сапёр» emMines

emMines является одной из первых попыток создания 3D-сапёра, сразу же заслужившей внимание со стороны пользователей. Как и в случае с Xbomb, фактическое уменьшение количества соседних ячеек в большинстве других трёхмерных версиях игры привели к небольшому снижению сложности. Практически ни одна версия игры не учитывала мины, расположенные в диагональных ячейках, остановившись лишь на непосредственно соседних узлах. Однако в emMines количество смежных ячеек могло достигать 26, что серьёзно усложняло игру для пользователя, что приводило к востребованности игры только среди опытных игроков, для которых поле стандартного сапёра становилось предсказуемым (мировой рекорд решения поля сложности «Опытный» составляет всего 7 секунд). В целом, emMines является попыткой перехода классического «Сапёра» в становившееся тогда популярным для игр трёхмерное пространство. С точки зрения же программиста программное средство практически не отличается от классического «Сапёра», а третье измерения в большинстве случаев заключается в добавлении ещё одного уровня цикла, создании трёхмерного массива вместо двумерного. Из интересного можно отметить появление полноценного «обучения» в данной версии игры.

В целом, можно сказать, что все версии игр имели схожий базовый функционал, а также одинаковые правила, что позволяет чётко сформулировать требования к будущему программному средству.

1.2 Выбор программной среды разработки

Реализация проекта планируется с использованием стандартной библиотеки Win32API (WinAPI), вероятно использовавшейся при написании оригинальных версий для Windows. Использовании языка C++ в сочетании с базовым набором функций и интерфейсов позволяет научиться писать приложения ОС Windows без использования дополнительных средств разработки. Так же выбор обусловлен желанием попробовать свои силы в разработке на языке C++, а также исправить ошибки прошлого опыта разработки игры с помощью WinAPI (приложение было реализовано без использования события WM_PAINT и в целом было примером того, как писать код не следует).

Также для разработки и отладки была выбрана среда программирования Microsoft Visual Studio 2019. Сочетание языка программирования, стандартных библиотек и средств среды разработки должно решить все возникающие во время написания программного средства задачи. Для работы со звуком доступна стандартная библиотека winmm.lib, для работы с файлами можно использовать один из многих стандартных файлов библиотеки языков C и C++, например, fstream или iostream. Более комфортную работу со строками по сравнению с «чистым» Си обеспечит класс string в C++, однако следует учитывать, что Win32 API не рассчитана на работу со строками, а потому придётся использовать стандартный функционал преобразования строк в массив символов. После окончания разработки среда Visual Studio предоставляет широкие возможности для отладки, а встроенные средства помощи разработчики могут подсказать не только где в коде закралась ошибка, но и возможные пути её решения. Так же можно воспользоваться стандартными средствами для создания установочного файла, а встроенный компоновщик позволит всего в несколько кликов назначить нужные зависимости, например, файлы музыкального сопровождения или графических примитивов для использования в программе.

В целом, использования развитого языка C++ с широкой библиотекой вспомогательного кода, а также продуманной и мощной среды разработки позволяет значительно упростить процесс разработки, сосредоточившись на обучении действительно важным вещам.

2 Постановка задачи

В рамках курсового проекта необходимо реализовать программное игровое приложение «MineSweeper». Создать игру необходимо в жанре «Головоломка», также выбран стиль 2D. При запуске приложения пользователь должен иметь возможность продолжить незаконченную предыдущую игру, увидеть статистику своих предыдущих игр или выбрать сложность следующей игры из представленных стандартных. Реализация возможности самостоятельного выбора размера поля сочтена излишней, поскольку стандартные варианты уже являются оптимизированными по сложности и позволяют решать поле в режиме соревнования – на время, в том числе и борясь за места в мировой таблице лидеров.

Плоское игровое поле разделено на смежные ячейки квадратной формы (классический вариант), в качестве графической составляющей выбран классический вариант Windows XP. Некоторые из ячеек «заминированы», их количество заранее известно. Минирование ячеек происходит после открытия первой ячейки, что не допускает возможности проигрыша первым же ходом.

Если под открытой ячейкой мины нет, то в ней появляется число, показывающее, сколько ячеек, соседствующих с только что открытой, «заминировано»; используя эти числа, игрок пытается рассчитать расположение мин. Если под соседними ячейками тоже нет мин, то открывается некоторая «не заминированная» область до ячеек, в которых есть цифры. «Заминированные» ячейки игрок может пометить, чтобы случайно не открыть их. Так же существует возможность пометить ячейку знаком «Вопрос» с целью привлечь своё внимание к ней в будущем.

Главная цель игры – открыть все ячейки поля, не содержащие бомб, заблокировав(пометив) при этом ячейки, в которых расположены бомбы.

Однако иногда даже в середине и в конце игры некоторые ячейки всё же приходится открывать наугад. Для решения данной проблемы и облегчения задачи игрока в игре должна быть реализована возможность победы без открытия всех ячеек. Если все мины помечены «Флажком», либо часть из них помечена так же «Вопросом», однако «Вопросами» помечены только ячейки с минами, это так же считается условием победы. Это позволяет в спорных ситуациях поочерёдно попробовать различные варианты решения, а победа в игре перебором всех ячеек считается невозможной.

В рамках проектирования приложения необходимо так же предусмотреть возможность дальнейшего расширения функциональности. Следует продумать возможность дальнейшего добавления возможностей, от которых на раннем этапе решено было отказаться.

3 Разработка программного средства

3.1 Проектирование класса игровой ситуации

Для хранения информации о игровой сессии было решено воспользоваться возможностью языка C++ по созданию классов. Был создан класс GameSession, хранящий в себе всю информации о текущей игровой ситуации. Для этого в классе были реализованы следующие поля:

```
int width;
int height;
int mines;
int steps = 0;
bool placed = false;
int** Real;
int** Field;
```

Поля width и height типа integer хранят информацию о размерности игрового поля, поле mines – количество мин. Поле steps отражает количество совершённых игроком ходов, а флаг placed – показывает, был ли совершён первый ход (открыта ячейка, после которой происходит расстановка мин). Два динамических массива типа integer хранят в себе информацию о текущей игровой сессии.

Использование класса для хранения информации о игровой сессии позволило упростить процедуры сохранения и загрузки информации о игре, сделало взаимодействие различных компонентов программы более наглядным, а также позволило вынести весь функционал игры сапёр в методы данного класса. Далее будут рассмотрены различные алгоритмы игры.

3.2 Проектирование алгоритма размещения мин

Размещение мин было реализовано с помощью метода класса GameSession Place (см. Приложение А).

Метод принимает в качестве параметров две координаты – ячейку, выбранную игроком для открытия. В дальнейшем программа начинает выполнение цикла, пока количество размещённых мин не станет равно заданному для текущей игровой сессии (поле mines). В цикле программа получает два случайных числа и проверяет соблюдение нескольких условий. Если все условия соблюдаются, в полученную ячейку ставиться мина, а счётчик установленных мин увеличивается. Среди условий следует отдельно выделить ограничение на положение мины. Кроме стандартных (мина уже установлена в данную ячейку, данная ячейка выбрана игроком) было также добавлено ограничение на близость мины к выбранной ячейки.

Сперва планировалось располагать мины как минимум в соседних ячейках для поля площадью не больше ста ячеек (минимальные уровни сложности) и на расстоянии одной клетки для остальных. Однако в результате тестирования было решено руководствоваться вторым условием независимо от размеров поля, таким образом первое нажатие даже на самых маленьких полях гарантирует открытие области, а не цифровой ячейки, минимизируя тем самым возможность проигрыша первым же ходом из-за «неудачно» открытой ячейки.

Обратной стороной является усложнение расстановки мин, в случае дальнейшего добавления возможности пользователя задания собственных параметров игры это может привести к нестабильной работе данного алгоритма, а также к размещению «сплошных» минных полей. Это лишний раз доказывает нецелесообразность задания пользовательских параметров минного поля.

3.3 Проектирование алгоритма открытия областей

Открытие ячеек в случае нахождения в ячейке нуля (в соседних ячейках нет мин) вызывает алгоритм открытия областей. Данный алгоритм также реализован в качестве метода класса GameSession и принимает те же параметры, что и предыдущий. Данный метод последовательно проходит по всем окрестным ячейкам и открывает их. Соответственно, если открытая ячейка так же является пустой, происходит рекурсивный вызов алгоритма, пока не будет достигнута граница области. Отсутствие ошибок гарантируется наличием двух массивов, таким образом можно контролировать, была ли уже открыта та или иная ячейка и не допускать переполнения стека при замкнутом рекурсивном вызове. В предыдущей версии игры вместо рекурсивного алгоритма использовался цикл, однако при сложной форме областей (типа буквы U или других выпуклых форм) алгоритм мог давать сбои. Применение рекурсии ускорило работу и решило возникавшие проблемы ценой незначительного увеличения затрат памяти. Теоретически на компьютерах с недостатком оперативной памяти на больших полях могут возникать проблемы с работой алгоритма, но это лишь лишний раз доказывает нецелесообразность создания пользовательских полей.

3.4 Проектирование алгоритма подсчёта количества шагов

Данный алгоритм добавлялся уже после реализации основной механики игры, а потому является добавлением в код уже существующих методов. Суть заключается в простой инкрементации поля класса при совершении игроком действий. Единственным действием, не инкрементирующим счётчик (помимо заведомо ложных наподобие повторного открытия открытой ячейки), является размещения знака «Вопроса», которое происходит после размещения «Флага»

при нажатии на ту же кнопку над той же ячейкой. Теоретически, возможна ситуация, когда между размещением «Флага» и «Вопроса» пользователь совершает другие действия, однако в целом в оригинальных играх редко реализуется подсчёт количества кликов мышкой или шагов, а потому для соревновательных целей это непринципиально.

3.5 Проектирование алгоритма проверки на победу игрока

Если реализация проверки на поражение является предельно понятной (открытие ячейки с миной), то при проверке на победу было решено реализовать дополнительные условия. Для этого в алгоритм были введены дополнительно счётчик установленных флагов и проверка на установку вопроса в неправильном месте. Таким образом алгоритм последовательно проверяет несколько условий:

Существование мины, не отмеченной ни «Флагом», ни знаком «Вопрос»

Установка «Флага» в неположенном месте.

Количество установленных «Флагов» равно количеству мин, либо не существует лишних знаков «Вопрос». Данное условие тесно связано с результатами предыдущих двух, и в сумме они гарантируют правильность результатов работы алгоритма, что было проверено тестами.

3.6 Проектирование алгоритмов сохранения и загрузки игрового состояния

Сохранение и загрузка в данном программном средстве применяются в двух ситуациях: при сохранении игровой статистики и сохранении текущей игровой сессии.

Сохранение сессии происходит после каждого действия игрока, включая создание поля. Удаляется же файл после победы или поражения на данном поле. Таким образом, на любом этапе игры игрок может спокойно закрыть приложение, затем продолжив с того же места при следующем запуске. Переигрывание же не допускается, однако «опытные» пользователи могут обнаружить файл сохранения игры и создать его резервную копию, что, в целом, является нормальной практикой даже для многих современных игр.

Сохранение и загрузка реализованы с помощью потоков fstream, направленных в файл. Данный класс обеспечивает возможность сохранения и загрузки объекта класса целиком, что позволяет легко и быстро расширять функционал класса дополнительными полями без необходимости модификации кода алгоритмов. При реализации загрузки дополнительно предусмотрена возможность отсутствия файла или существования пустого файла (хотя сама программа этого не допускает).

3.7 Проектирование алгоритмов хранения, сохранения и загрузки игровой статистики

Для хранения игровой статистики были дополнительно реализованы класс GameStat и структура GameStats, необходимые для реализации односвязного динамического списка. Использование динамической структуры данных позволяет неограниченно увеличивать количество хранимых игровых сессий, а так же упрощает работу с ними. Класс же был создан для более простой работы с данными класса GameSession, некоторые из которых после окончания игры более не важны. Сохранение и загрузка выполнены аналогичным с игровой сессией образом, разница наблюдается лишь в последовательном сохранении в динамическую структуру данных с помощью цикла. По окончании загрузки указатель остаётся на последнем элементе очереди, что позволяет продолжить запись результатов игр, что было проверено во время тестирования.

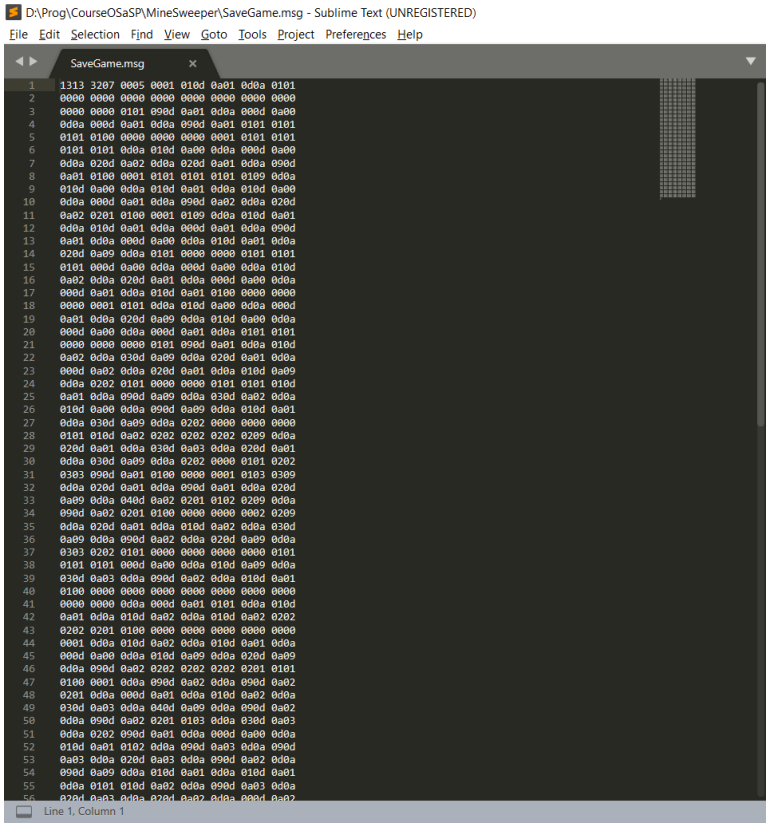

Рисунок 3.1 – Файл сохранения игры (сложность «Выбор команды»)

4 Тестирование программного средства

После написания программы необходимо сделать проверку различных функций и ситуаций в программе, для убеждения в том, что программа работает исправно. Далее будут рассмотрены различные тестовые ситуации, в которых ожидалось возникновение проблем в ходе эксплуатации программы.

Тестирование производилось на персональном компьютере с установленной операционной системой Windows 10, в режимах Debug и Release, а также после установки программы с помощью установщика.

Так же тестирование проводилось на другом устройстве, где после установки необходимых библиотек (Распространяемые компоненты Microsoft Visual C++) приложение так же прошло все тесты.

4.1 Тестирование функционала программы

Таблица 4.1 – Тестирование функционала программы

Номер теста	Тестируемая функциональность	Ожидаемый и полученный результаты
1	Открытие области при первом клике пользователя	В первой версии игры на небольших размерах поля часто открывались ячейки-цифры, после обновления работа стабильна
2	Повторные клики по открытым ячейкам	Программа правильно обрабатывает клики, алгоритм открытия не изменяет ни содержимое ячеек, ни счётчик шагов. При изменении содержимого ячейки в файле сохранения (манипуляции «опытного» пользователя) программа так же работает правильно.
3	Попытка открытия второй области	Результаты остаются непредсказуемыми, нельзя гарантированно сказать, будет ли открыта мина, область или ячейка-цифра
4	Попытка обозначения мины	В случае открытой ячейки программа вызывает алгоритм, который не изменяет ячейку. В противном случае происходит

Продолжение таблицы 4.1

		отметка ячейки «Флажком» и изменение счётчика шагов.
9	Попытка обозначения «Вопроса»	После повторного щелчка программа выставляет новое значение ячейки, счётчик шагов не изменяется
10	Повторное изменение «Вопроса» на «Флаг»	Алгоритм реагирует адекватно, после повторного нажатия вновь возникает «Флаг», счётчик инкрементируется
11	Попытка открытия «Флага»	Алгоритм реагирует адекватно, значение ячейки не изменяется
12	Попытка открытия «Вопроса»	Алгоритм реагирует адекватно, ячейка открывается либо происходит взрыв мин, проигрыш игры в зависимости от содержимой ячейки
13	Открытие всех ячеек	Победа в игре
14	Обозначение всех мин «Флажками»	Победа в игре
15	Обозначение всех мин «Вопросами»	Победа в игре
16	Обозначение всех мин «Флажками» и «Вопросами» (без лишних обозначений)	Победа в игре
17	Обозначение всех мин «Флажками» и «Вопросами» (с лишними обозначениями)	Продолжение игры
18	Победа в игре	Удаление файла сохранения игры, перезапись файла статистики
19	Преждевременный выход из игры	Текущая игра остаётся в сохранении
20	Повторный запуск игры (после незаконченной сессии)	Игра продолжается с момента сохранения
21	Повторный запуск игры (после окончания сессии)	Выводится статистика игрока
22	Запуск игры без файлов сохранения и статистики	Выводится окно начала новой игры

Продолжение таблицы 4.1

23	Контроль времени игры (<60 секунд)	Правильное отображение
24	Контроль времени игры (>1 минуты)	Правильное отображение
25	Проверка правильности загрузки данных о времени игры	Правильное отображение
26	Проверка правильности отображения информации о поставленных «Флажках»	Правильное отображение
27	Проверка правильности загрузки данных о выставленных «Флажках»	Правильное отображение

4.2 Выводы по прохождению тестирования

Приложение успешно прошло все тесты, что показывает корректность работы разработанного программного средства и соответствие функциональным требованиям. Работа с файлами и звуком так же проходили в штатном режиме.

Во время тестирования была обнаружена недоработка, связанная с победой в случае открытия ячейки с миной до их расстановки (мина первым ходом). Данная проблема была исправлена, в дальнейшем приложение работало в штатном режиме.

Также были обнаружены проблемы с открытием файла статистики из только что загруженной игры. Загрузка файла статистики при этом не происходила, игроку отображалась пустая статистика, а после победы предыдущие записи удалялись. Данная проблема так же была решена.

5 Руководство пользователя программного средства

5.1 Системные требования

Для нормальной работы программного средства необходимы следующие минимальные системные требования:

– операционная система: Windows 10, Windows 8.1, Windows 7, Windows XP;

– процессор: с тактовой частотой 1Ггц;

– оперативная память 256 МБ;

– свободное место на жестком диске: 16 МБ.

5.2 Установка

Установка используется с помощью стандартных установочных файлов (setup.exe, Setup.msi). Инструкции в установочных файлах позволяют без труда установить программу даже неопытным пользователям ПК, программа имеет возможность установки в папку по выбору пользователя, создаёт ярлык на рабочем столе. При необходимости игру можно так же удалить с помощью установщика. На любом этапе установки можно вернуться назад.

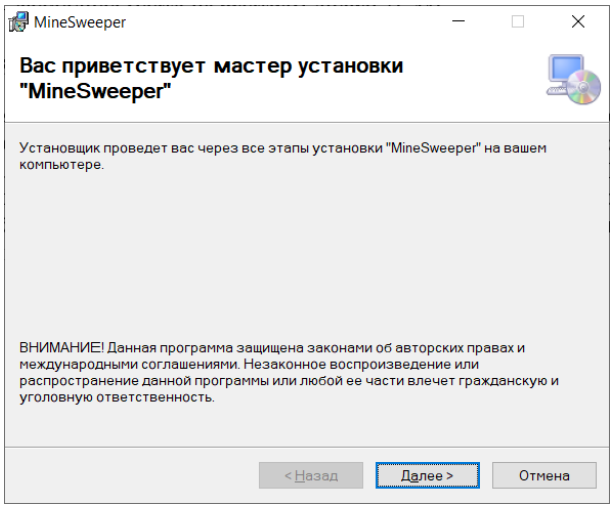

Рисунок 5.1 – Стартовое окно установки

На следующем скриншоте продемонстрирована возможность установки игры в выбранную пользователем папку любого диска.

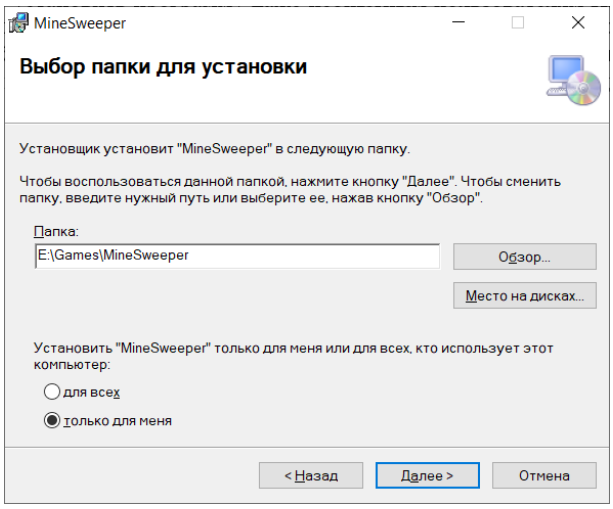

Рисунок 5.2 – Окно выбора папки для установки

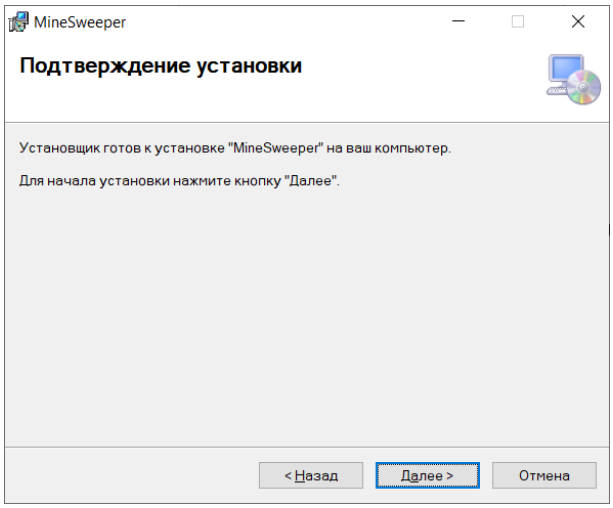

Рисунок 5.3 – Окно подтверждения установки

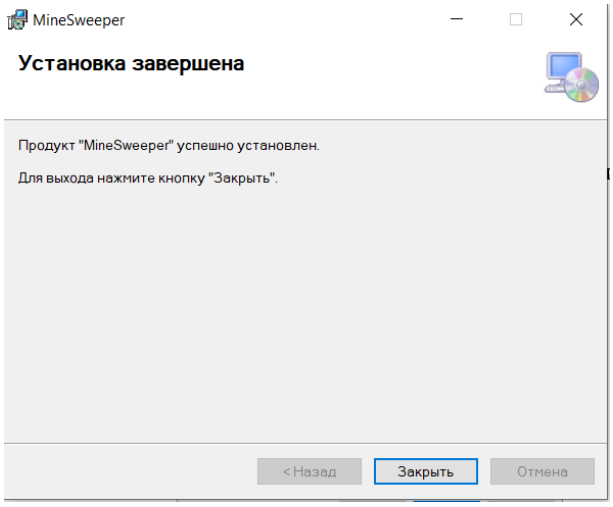

Рисунок 5.4 – Игра установлена

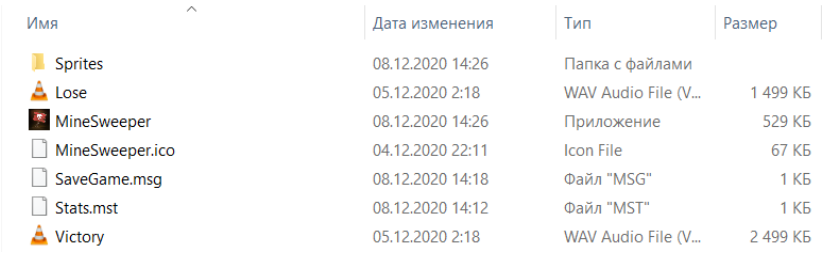

Рисунок 5.5 – Содержимое каталога с установленной игрой

5.3 Использование установленной программы

Запуск игры возможен с помощью ярлыка в папке с установленным приложением или на рабочем столе.

Рисунок 5.6 – Ярлык игры на рабочем столе

После первого запуска игрок попадает на экран «Новая игра», где может выбрать желаемую сложность.

Рисунок 5.7 – Окно выбора сложности игры

Нажатием на одну из кнопок игрок выбирает желаемую сложность и попадает на игровое поле, где может выбрать одну из ячеек (рис. 5.8). Нажатие левой кнопки мыши на любую из ячеек открывает игровую область, после чего возможно начало решения головоломки.

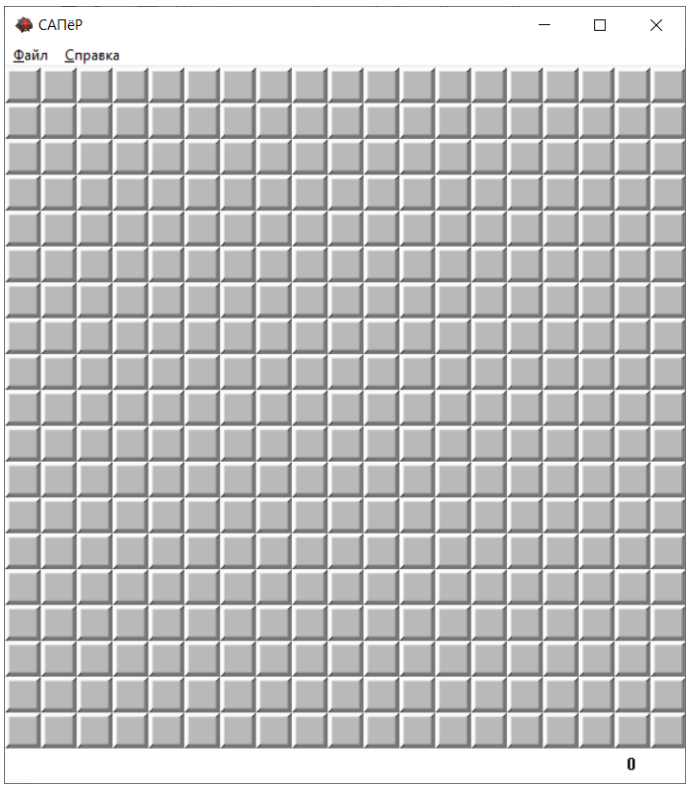

Рисунок 5.8 – Начало игры на сложности «Выбор команды»

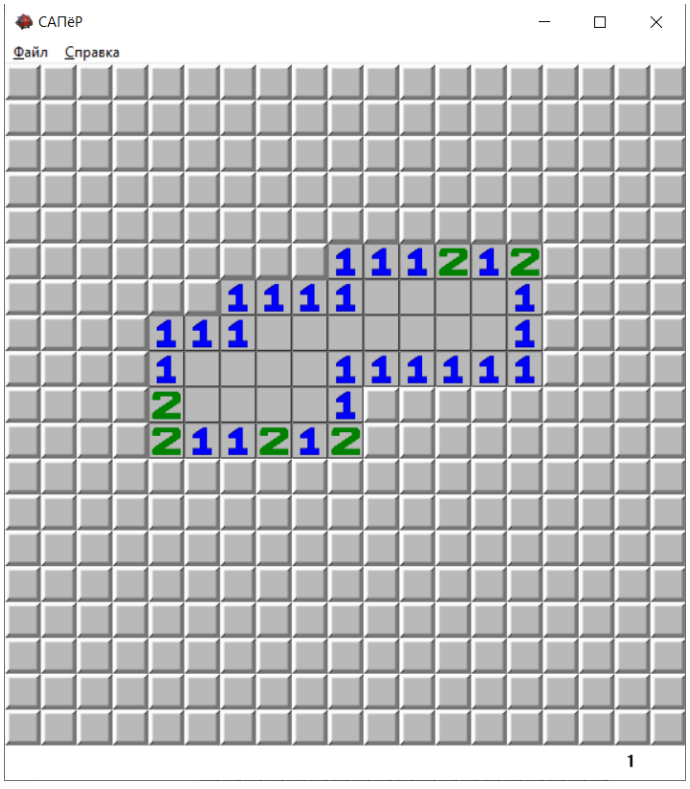

Рисунок 5.8 – Первый ход на сложности «Выбор команды»

В дальнейшем игрок продолжает решение головоломки, открывая новые области и ячейки и помечая мины и знаки «Вопрос». Счётчик шагов в правой нижней части экрана показывает количество совершённых ходов.

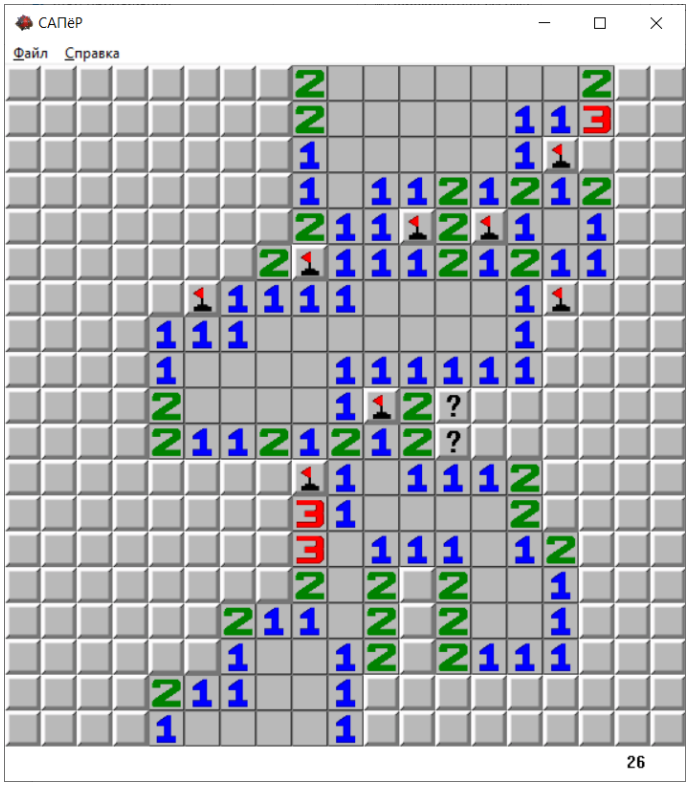

Рисунок 5.9 – Игра на сложности «Выбор команды»

После победы пользователь теряет возможность изменения поля, проигрывается звук победы. То же происходит и при открытии мины. Дополнительно выводится соответствующее текстовое сообщение.

В любой момент игрок может начать новую сессию через меню Файл -> Новая игра -> *Выбранная сложность*

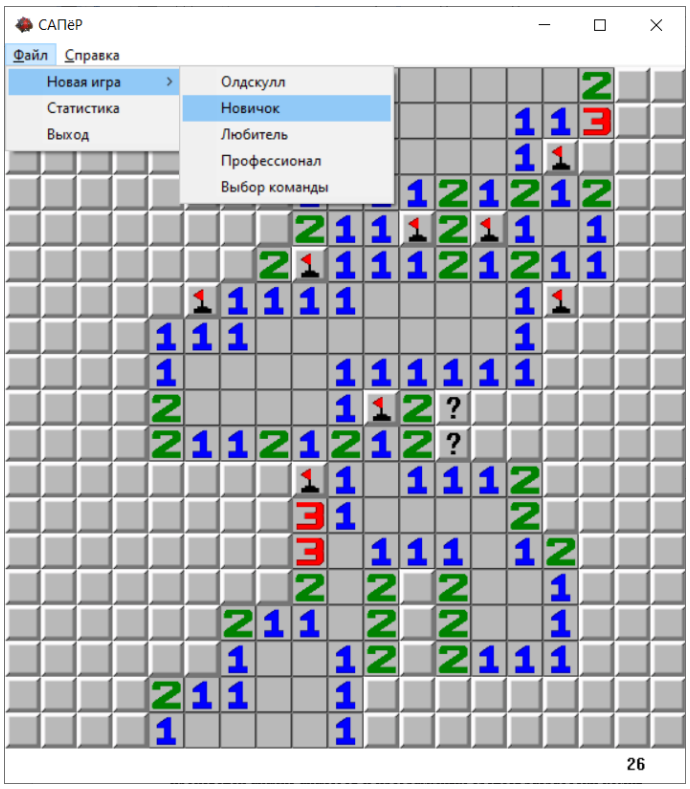

Рисунок 5.10 – Выбор сложности новой игры

ЗАКЛЮЧЕНИЕ

В рамках курсовой работы была изучена и проработана технология разработки игр путем создания игрового приложения «MineSweeper» в жанре «Головоломка». Разработка велась на языке программирования C++ с использованием стандартной библиотеки WinAPI и других вспомогательных стандартных системных библиотек. В сравнении с аналогами данная игра имеет весь базовый функционал, а также звуковое сопровождение, статистику и возможность прерывания и продолжения игры.

Для достижения данной цели были решены следующие задачи:

– осуществлена постановка игровой задачи;

– произведен анализ аналогов и программных средств разработки компьютерной игры;

– произведен поиск и подбор бесплатных моделей;

– повторены и углублены знания в языках программирования C и C++;

– изучена работа с платформой Windows, средой разработки MS Visual Studio 2019;

– спроектировано приложение;

– изучена возможность создания установочных файлов в среде MS VS;

– составлены руководство пользователя и документация.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Профессии будущего. [Электронный ресурс]. – Режим доступа: http://www.proforientator.ru/publications/articles/detail.php?ID=5454

[2] Microsoft Documentation [Электронный ресурс] Режим доступа: https://docs.microsoft.com/en-us/windows/win32/ – Дата доступа 28.10.2020.

[3] Основы программирования для Win32 API [Электронный ресурс] Режим доступа: https://dims.karelia.ru/win32/ – Дата доступа 20.11.2020.

[4] Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста – СПб.: Питер, 2013. – 464 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5–496-00487-9.

[5] MSDN – Windows API по-русски [Электронный ресурс] Режим доступа: http://narovol.narod.ru/_tbkp/New_MSDN_API/index_msdn.htm – Дата доступа 29.10.2020.

[6] Win 32 API по шагам [Электронный ресурс] Режим доступа: https://www.firststeps.ru/ – Дата доступа 02.11.2020.

[7] Справочник по функциям Windows API [Электронный ресурс] Режим доступа: http://rusproject.narod.ru/winapi/winapi.htm – Дата доступа 02.11.2020.

ПРИЛОЖЕНИЕ А
(Обязательное)
Исходный код программы

```
#include "pch.h"
#include "framework.h"
#include "MineSweeper.h"
using namespace std;

#define SavePath     "SaveGame.msg"
#define StatsPath    "Stats.mst"
#define MineWidth    30

HINSTANCE hInst;
bool playing = false;
bool DrawingStats = false;
bool NewGame = false;

HWND hButtonClassic;
HWND hButtonNewbie;
HWND hButtonAmateur;
HWND hButtonExperienced;
HWND hButtonMySelection;

HBITMAP MS0;
HBITMAP MS1;
HBITMAP MS2;
HBITMAP MS3;
HBITMAP MS4;
HBITMAP MS5;
HBITMAP MS6;
HBITMAP MS7;
HBITMAP MS8;
HBITMAP MS9;
HBITMAP MS10;
HBITMAP MS11;
HBITMAP MS13;
HBITMAP MS15;

class GameSession
{
public:
    int width;
    int height;
    int mines;
    int steps = 0;
    int timeM = 0;
    int timeS = 0;
    int flags = 0;
    bool placed = false;
    int** Real;
    int** Field;

    GameSession()
    {
        width = 0;
        height = 0;
        mines = 0;
        Field = NULL;
        Real = NULL;
    }
    GameSession(int x, int y, int m)
```

```
    {
        mines = m;
        width = x;
        height = y;
        Real = new int*[height];
        Field = new int* [height];
        for (int line = 0; line < height; line++)
        {
            Real[line] = new int[width];
            Field[line] = new int[width];
            for (int col = 0; col < width; col++)
            {
                Real[line][col] = 0;
                Field[line][col] = 10;
            }
        }
    }

    void Place(int X, int Y)
    {
        int mine = 0;
        while (mine < mines)
        {
            int posX = rand() % width;
            int posY = rand() % height;
            if (abs(posX - X) > 1 || abs(posY - Y) > 1) //((width*height <= 100 && posX
!= X && posY != Y) || (width * height > 100 && (abs(posX-X)>1 || abs(posY - Y) > 1)))
            {
                if (Real[posY][posX] != 9)
                {
                    Real[posY][posX] = 9;
                    mine++;
                }
            }
        }
        placed = true;
    }

    void Count()
    {
        for (int y = 0; y < height; y++)
        for (int x = 0; x < width; x++)
            if (Real[y][x] != 9)
                for (int dx = -1; dx < 2; dx++)
                for (int dy = -1; dy < 2; dy++)
                    if (x + dx >= 0 && x + dx < width && y + dy >= 0 && y + dy < height)
                        if (Real[y + dy][x + dx] == 9)
                        {
                            Real[y][x] = Real[y][x]++;
                        }
    }

    bool Defuse(int X, int Y)
    {
        if (!placed)
        {
            Place(X, Y);
            Count();
        }
        if (Field[Y][X] != 15)
        {
            if (Real[Y][X] == 9)
            {
                Field[Y][X] = 13;
```

```
                steps++;
                return false;
            }
            else
            {
                if (Field[Y][X] != Real[Y][X])
                {
                    Field[Y][X] = Real[Y][X];
                    steps++;
                    if (Real[Y][X] == 0)
                        Open(X, Y);
                    return true;
                }
            }
        }
        return true;
    }

    void Open(int X, int Y)
    {
        for (int dx = -1; dx < 2; dx++)
        for (int dy = -1; dy < 2; dy++)
            if (X + dx >= 0 && X + dx < width && Y + dy >= 0 && Y + dy < height && !(dx
== 0 && dy == 0))
            {
                if (Real[Y + dy][X + dx] == 0 && Field[Y + dy][X + dx] != 0)
                {
                    Field[Y + dy][X + dx] = Real[Y + dy][X + dx];
                    Open(X + dx, Y + dy);
                }
                else
                    Field[Y + dy][X + dx] = Real[Y + dy][X + dx];
            }
    }

    void Mark(int X, int Y)
    {
        if (placed)
            switch (Field[Y][X])
            {
            case 10:
                Field[Y][X] = 15;
                flags++;
                steps++;
                break;
            case 11:
                Field[Y][X] = 10;
                break;
            case 15:
                Field[Y][X] = 11;
                flags--;
                break;
            }
    }

    bool NotWinState()
    {
        if (!placed)
            return true;
        // int flags = 0;
        bool wrfl = false;
        for (int y = 0; y < height; y++)
        for (int x = 0; x < width; x++)
        {
```

```
            if (Real[y][x] == 9 && !(Field[y][x] == 11 || Field[y][x] == 15))
                return true;
            if (Real[y][x] != 9 && Field[y][x] == 15)
                return true;
            /*if (Field[y][x] == 15)
                flags++;*/
            if (Field[y][x] == 11 && Real[y][x] != 9)
                wrfl = true;
        }
        if (flags == mines || !wrfl)
            return false;
        return true;
    }

    void Save()
    {
        fstream SF;
        SF.open(SavePath, fstream::out | fstream::trunc);
        if (SF.is_open())
        {
            SF.put((char)width);
            SF.put((char)height);
            SF.put((char)mines);
            SF.put((char)steps);
            SF.put((char)timeM);
            SF.put((char)timeS);
            SF.put((char)flags);
            SF.put((char)placed);

            for (int line = 0; line < height; line++)
                for (int col = 0; col < width; col++)
                {
                    SF.put((char)Real[line][col]);
                    SF.put((char)Field[line][col]);
                }
        }
        SF.close();
    }

    bool Load()
    {
        fstream SF;
        SF.open(SavePath, fstream::in);
        if (SF.is_open())
        {
            if (SF.peek() == std::ifstream::traits_type::eof())
            {
                SF.close();
                return false;
            }
            else
            {
                width = SF.get();
                height = SF.get();
                mines = SF.get();
                steps = SF.get();
                timeM = SF.get();
                timeS = SF.get();
                flags = SF.get();
                placed = SF.get();

                Real = new int* [height];
                Field = new int* [height];
                for (int line = 0; line < height; line++)
```

```
                {
                    Real[line] = new int[width];
                    Field[line] = new int[width];
                }

                for (int line = 0; line < height; line++)
                    for (int col = 0; col < width; col++)
                    {
                        Real[line][col] = SF.get();
                        Field[line][col] = SF.get();
                    }
                SF.close();
                return true;
            }
        }
        SF.close();
        return false;
    }
};

class GameStat
{
public:
    int width;
    int height;
    int mines;
    int steps;
    int timeM;
    int timeS;

    GameStat()
    {
        width = height = mines = steps = timeM = timeS = 0;
    }
    GameStat(GameSession Game)
    {
        width = Game.width;
        height = Game.height;
        mines = Game.mines;
        steps = Game.steps;
        timeM = Game.timeM;
        timeS = Game.timeS;
    }
};

struct GameStats
{
    GameStat stat;
    GameStats* next = NULL;

    void Save()
    {
        fstream SF;
        SF.open(StatsPath, fstream::out | fstream::trunc);
        if (SF.is_open() && (this != NULL))
        {
            GameStats* Cur = this;
            do
            {
                SF.put((char)(*Cur).stat.width);
                SF.put((char)(*Cur).stat.height);
                SF.put((char)(*Cur).stat.mines);
                SF.put((char)(*Cur).stat.steps);
                SF.put((char)(*Cur).stat.timeM);
```

```
                SF.put((char)(*Cur).stat.timeS);
                Cur = (*Cur).next;
            } while (Cur != NULL);
        }
        SF.close();
    }
};

GameSession Game;

GameStats* Stats = NULL;
GameStats* CurStat = NULL;

ATOM                MyRegisterClass(HINSTANCE hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);
void                DrawGrid(HDC);
void                DrawStats(HDC);
void                DrawNew();
void                DestroyNew();
void                WinGame(HWND);
void                LoseGame(HWND);
bool                LoadStats();


int APIENTRY wWinMain(HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPWSTR    lpCmdLine,
                      int       nCmdShow)
{
    MyRegisterClass(hInstance);

    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }

    MSG msg;
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return (int) msg.wParam;
}

ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEXW wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style          = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc    = WndProc;
    wcex.cbClsExtra     = 0;
    wcex.cbWndExtra     = 0;
    wcex.hInstance      = hInstance;
    wcex.hIcon          = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_MINESWEEPER));
    wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));
    wcex.hCursor        = LoadCursor(nullptr, IDC_ARROW);
    wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
    wcex.lpszMenuName   = MAKEINTRESOURCEW(IDC_MINESWEEPER);
```

```
    wcex.lpszClassName  = L"MainWNDW";

    return RegisterClassExW(&wcex);
}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
   int posX, posY, width, height;
   if (Game.Load())
   {
       width = Game.width * MineWidth;
       height = Game.height * MineWidth;
       posX = (GetSystemMetrics(SM_CXSCREEN) - width) / 2;
       posY = (GetSystemMetrics(SM_CYSCREEN) - height) / 2;
       playing = true;
   }
   else
   {
       posX = CW_USEDEFAULT;
       posY = 0;
       width = 250;
       height = 350;

       Stats = NULL;
       CurStat = Stats;
       if (LoadStats())
       {
           DrawingStats = true;
       }
       else
       {
           NewGame = true;
       }
   }

   HWND hWnd = CreateWindowW(L"MainWNDW",L"САПёР", WS_OVERLAPPEDWINDOW,
                             posX, posY, width, height,
                             nullptr, nullptr, hInstance, nullptr);

   if (!hWnd)
   {
      return FALSE;
   }

   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);

   return TRUE;
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int xPos;
    int yPos;
    RECT Rect;

    switch (message)
    {
    case WM_CREATE:
        MS0  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_0.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS1  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_1.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS2  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_2.bmp", IMAGE_BITMAP, 0, 0,
```

```
LR_LOADFROMFILE);
        MS3  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_3.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS4  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_4.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS5  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_5.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS6  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_6.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS7  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_7.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS8  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_8.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS9  = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_mine.bmp", IMAGE_BITMAP, 0,
0, LR_LOADFROMFILE);
        MS10 = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_10.bmp", IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE);
        MS11 = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_question.bmp", IMAGE_BITMAP,
0, 0, LR_LOADFROMFILE);
        MS13 = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_mine.bmp", IMAGE_BITMAP, 0,
0, LR_LOADFROMFILE);
        MS15 = (HBITMAP)LoadImageA(NULL, "Sprites/Minesweeper_flag.bmp", IMAGE_BITMAP, 0,
0, LR_LOADFROMFILE);
        hButtonClassic = CreateWindowW(L"BUTTON", L"Олдскул", WS_CHILD | BS_PUSHBUTTON,
            25, 25, 175, 50, hWnd, (HMENU)IDM_Classic, hInst, NULL);
        hButtonNewbie = CreateWindowW(L"BUTTON", L"Новичёк", WS_CHILD | BS_PUSHBUTTON,
            25, 75, 175, 50, hWnd, (HMENU)IDM_Newbie, hInst, NULL);
        hButtonAmateur = CreateWindowW(L"BUTTON", L"Любитель", WS_CHILD | BS_PUSHBUTTON,
            25, 125, 175, 50, hWnd, (HMENU)IDM_Amateur, hInst, NULL);
        hButtonExperienced = CreateWindowW(L"BUTTON", L"Опытный", WS_CHILD | BS_PUSHBUTTON,
            25, 175, 175, 50, hWnd, (HMENU)IDM_Classic, hInst, NULL);
        hButtonMySelection  =  CreateWindowW(L"BUTTON",  L"Выбор  команды",  WS_CHILD  |
BS_PUSHBUTTON,
            25, 225, 175, 50, hWnd, (HMENU)IDM_MySelection, hInst, NULL);
        SetTimer(hWnd, IDT_TIMER, 1000, NULL);
        break;
    case WM_LBUTTONDOWN:
    case WM_RBUTTONDOWN:
        if (playing)
        {
            xPos = LOWORD(lParam) / MineWidth;
            yPos = HIWORD(lParam) / MineWidth;
            if (message == WM_LBUTTONDOWN)
                playing = Game.Defuse(xPos, yPos);
            else
                Game.Mark(xPos, yPos);
            InvalidateRect(hWnd, NULL, true);
            if (playing)
            {
                playing = Game.NotWinState();
                if (!playing)
                    WinGame(hWnd);
                else
                    Game.Save();
            }
            else
                LoseGame(hWnd);
        }
        break;
    case WM_COMMAND:
        {
            DestroyNew();
            int wmId = LOWORD(wParam);
            switch (wmId)
```

```
            {
            case IDM_Continue:
                if (Game.Load())
                {
                    DestroyNew();
                    playing = true;
                    NewGame = false;
                    DrawingStats = false;
                    GetWindowRect(hWnd, &Rect);
                    MoveWindow(hWnd, Rect.left, Rect.top, MineWidth * Game.width + 15, Min-
eWidth * Game.height + 88, TRUE);
                    InvalidateRect(hWnd, NULL, true);
                }
                break;
            case IDM_Classic:
                DestroyNew();
                Game = GameSession(8, 8, 10);
                playing = true;
                NewGame = false;
                DrawingStats = false;
                GetWindowRect(hWnd, &Rect);
                MoveWindow(hWnd, Rect.left, Rect.top, MineWidth * Game.width + 15, Min-
eWidth * Game.height + 88, TRUE);
                InvalidateRect(hWnd, NULL, true);
                break;
            case IDM_Newbie:
                DestroyNew();
                Game = GameSession(9, 9, 10);
                playing = true;
                NewGame = false;
                DrawingStats = false;
                GetWindowRect(hWnd, &Rect);
                MoveWindow(hWnd, Rect.left, Rect.top, MineWidth * Game.width + 15, Min-
eWidth * Game.height + 88, TRUE);
                InvalidateRect(hWnd, NULL, true);
                break;
            case IDM_Amateur:
                DestroyNew();
                Game = GameSession(16, 16, 40);
                playing = true;
                NewGame = false;
                DrawingStats = false;
                GetWindowRect(hWnd, &Rect);
                MoveWindow(hWnd, Rect.left, Rect.top, MineWidth * Game.width + 15, Min-
eWidth * Game.height + 88, TRUE);
                InvalidateRect(hWnd, NULL, true);
                break;
            case IDM_Experienced:
                DestroyNew();
                Game = GameSession(30, 16, 99);
                playing = true;
                NewGame = false;
                DrawingStats = false;
                GetWindowRect(hWnd, &Rect);
                MoveWindow(hWnd, Rect.left, Rect.top, MineWidth * Game.width + 15, Min-
eWidth * Game.height + 88, TRUE);
                InvalidateRect(hWnd, NULL, true);
                break;
            case IDM_MySelection:
                DestroyNew();
                Game = GameSession(19, 19, 50);
                playing = true;
                NewGame = false;
                DrawingStats = false;
```

```
                GetWindowRect(hWnd, &Rect);
                MoveWindow(hWnd, Rect.left, Rect.top, MineWidth * Game.width + 15, Min-
eWidth * Game.height + 88, TRUE);
                InvalidateRect(hWnd, NULL, true);
                break;
            case IDM_STATS:
                if (playing)
                    Game.Save();
                DestroyNew();
                playing = false;
                NewGame = false;
                DrawingStats = true;
                GetWindowRect(hWnd, &Rect);
                MoveWindow(hWnd, Rect.left, Rect.top, 250, 350, TRUE);
                InvalidateRect(hWnd, NULL, true);
                break;
            case IDM_ABOUT:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                break;
            case IDM_EXIT:
                DestroyWindow(hWnd);
                break;
            default:
                return DefWindowProc(hWnd, message, wParam, lParam);
            }
        }
        break;
    case WM_PAINT:
        {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hWnd, &ps);
            if (DrawingStats)
            {
                DrawStats(hdc);
            }
            else if (NewGame)
            {
                DrawNew();
            }
            else
            {
                DrawGrid(hdc);
            }
            EndPaint(hWnd, &ps);
            UpdateWindow(hWnd);
        }
        break;
    case WM_SIZE:
        if (playing)
        {
            GetWindowRect(hWnd, &Rect);
            MoveWindow(hWnd, Rect.left, Rect.top, MineWidth * Game.width + 15, MineWidth *
Game.height + 88, TRUE);
        }
        break;
    case WM_DESTROY:
        if (playing)
            Game.Save();
        PostQuitMessage(0);
        break;
    case WM_TIMER:
        if (playing)
        {
            Game.timeS++;
```

```
            if (Game.timeS == 60)
            {
                Game.timeM++;
                Game.timeS = 0;
            }
            string time;
            if (Game.timeM > 0)
                time = to_string(Game.timeM) + ":";
            time += to_string(Game.timeS);
            RECT Rect;
            Rect.top = MineWidth * Game.height;
            Rect.left = 25;
            Rect.right = 25 + strlen(time.c_str()) * 8;
            Rect.bottom = Rect.top + 20;
            InvalidateRect(hWnd, &Rect, true);
        }
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}

INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;
    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}

void DrawGrid(HDC hdc)
{
    for (int y = 0; y < Game.height; y++)
    for (int x = 0; x < Game.width; x++)
    {
        HDC hCompatibleDC = CreateCompatibleDC(hdc);
        switch (Game.Field[y][x])
        {
        case 0:
            SelectObject(hCompatibleDC, MS0);
            break;
        case 1:
            SelectObject(hCompatibleDC, MS1);
            break;
        case 2:
            SelectObject(hCompatibleDC, MS2);
            break;
        case 3:
            SelectObject(hCompatibleDC, MS3);
            break;
        case 4:
            SelectObject(hCompatibleDC, MS4);
            break;
        case 5:
```

```
            SelectObject(hCompatibleDC, MS5);
            break;
        case 6:
            SelectObject(hCompatibleDC, MS6);
            break;
        case 7:
            SelectObject(hCompatibleDC, MS7);
            break;
        case 8:
            SelectObject(hCompatibleDC, MS8);
            break;
        case 9:
            SelectObject(hCompatibleDC, MS9);
            break;
        case 10:
            SelectObject(hCompatibleDC, MS10);
            break;
        case 11:
            SelectObject(hCompatibleDC, MS11);
            break;
        case 13:
            SelectObject(hCompatibleDC, MS13);
            break;
        case 15:
            SelectObject(hCompatibleDC, MS15);
            break;
        }
        StretchBlt(hdc, MineWidth * x, MineWidth * y, MineWidth, MineWidth, hCompati-
bleDC, 0, 0, 76, 76, SRCCOPY);
        DeleteObject(hCompatibleDC);
    }
    string time;
    if (Game.timeM > 0)
        time = to_string(Game.timeM) + ":";
    time += to_string(Game.timeS);
    TextOutA(hdc, 25, MineWidth* Game.height + 5, (time.c_str()), strlen(time.c_str()));
    string flString = to_string(Game.flags) + "/" + to_string(Game.mines);
    TextOutA(hdc, (MineWidth * Game.width - strlen(flString.c_str()) * 8) / 2, MineWidth
* Game.height + 5, (flString.c_str()), strlen(flString.c_str()));
    TextOutA(hdc, MineWidth * Game.width - 25 - strlen(to_string(Game.steps).c_str()) *
8, MineWidth * Game.height + 5, (to_string(Game.steps).c_str()),
strlen(to_string(Game.steps).c_str()));
}

void DrawStats(HDC hdc)
{
    RECT rect;
    rect.top = 0;
    rect.left = 10;
    rect.right = 230;
    rect.bottom = 40;

    GameStats* CurStats = Stats;
    while (CurStats != NULL)
    {
        string otp = to_string(CurStats->stat.width) + " x " + to_string(CurStats-
>stat.height) + " | " + to_string(CurStats->stat.mines) + " мин | " +
                     to_string(CurStats->stat.steps) + " шаг | " + to_string(CurStats-
>stat.timeM) + ":" + to_string(CurStats->stat.timeS);
        int D = DrawTextA(hdc, otp.c_str(), strlen(otp.c_str()), &rect, DT_NOCLIP ||
DT_CALCRECT);
        CurStats = CurStats->next;
        rect.top += D;
        rect.bottom += D;
```

```
    }
}

void DrawNew()
{
    ShowWindow(hButtonClassic, SW_SHOW);
    ShowWindow(hButtonNewbie, SW_SHOW);
    ShowWindow(hButtonAmateur, SW_SHOW);
    ShowWindow(hButtonExperienced, SW_SHOW);
    ShowWindow(hButtonMySelection, SW_SHOW);
}

void DestroyNew()
{
    ShowWindow(hButtonClassic, SW_HIDE);
    ShowWindow(hButtonNewbie, SW_HIDE);
    ShowWindow(hButtonAmateur, SW_HIDE);
    ShowWindow(hButtonExperienced, SW_HIDE);
    ShowWindow(hButtonMySelection, SW_HIDE);

}

void WinGame(HWND wnd)
{
    PlaySound(L"Victory.wav", NULL, SND_ASYNC | SND_FILENAME);
    DeleteFileA(SavePath);
    GameStat CurGame = GameStat(Game);
    if (Stats == NULL)
    {
        CurStat = new GameStats;
        CurStat->stat = CurGame;
        CurStat->next = NULL;
        Stats = CurStat;
    }
    else
    {
        GameStats* temp = new GameStats;
        temp->stat = CurGame;
        temp->next = NULL;
        CurStat->next = temp;
        CurStat = temp;
    }
    Stats->Save();
    MessageBoxA(wnd, "Поздравляем вас с решнием головоломки!", "Вы победили!", MB_OK);
}

void LoseGame(HWND wnd)
{
    PlaySound(L"Lose.wav", NULL, SND_ASYNC | SND_FILENAME);
    DeleteFileA(SavePath);
    MessageBoxA(wnd, "Не бойтесь попробовать свои силы ещё раз!", "Вас разорвало!",
MB_OK);
}

bool LoadStats()
{
    fstream SF;
    SF.open(StatsPath, fstream::in);
    if (SF.is_open())
    {
        if (SF.peek() == std::ifstream::traits_type::eof())
        {
            SF.close();
            return false;
```

```
        }
        else
        {
            while (SF.peek() != std::ifstream::traits_type::eof())
            {
                GameStat CurGame;
                char ch;
                SF.get(ch);
                CurGame.width = ch;
                SF.get(ch);
                CurGame.height = ch;
                SF.get(ch);
                CurGame.mines = ch;
                SF.get(ch);
                CurGame.steps = ch;
                SF.get(ch);
                CurGame.timeM = ch;
                SF.get(ch);
                CurGame.timeS = ch;

                if (Stats == NULL)
                {
                    Stats = new GameStats;
                    Stats->stat = CurGame;
                    Stats->next = NULL;
                    CurStat = Stats;
                }
                else
                {
                    GameStats* temp = new GameStats;
                    temp->stat = CurGame;
                    temp->next = NULL;
                    CurStat->next = temp;
                    CurStat = temp;
                }
            }
            return true;
        }
    }
    SF.close();
    return false;
}
```

ВЕДОМОСТЬ ДОКУМЕНТОВ

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1–40 01 01 613 ПЗ	Пояснительная записка	42
	Графические документы	
ГУИР.851006-013 СА	Схема работы алгоритма Open	Формат А1

					БГУИР КП I- 40 01 01 613 ПЗ		
Из	Лист	№ докум.	Подп.	Дата			
Разраб.		Крот Е.Д.			Программное средство Игра «MineSweeper» Ведомость курсового проекта	Литера: Т	Лист: 42 / Листов: 42
Провер.		Жиденко А.Л.				Кафедра ПОИТ гр. 851006	