СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc59474490)

[1 АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧ 5](#_Toc59474491)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ 7](#_Toc59474492)

[3 РЕАЛИЗАЦИЯ ПРОГРАММЫ 8](#_Toc59474493)

[4 ТЕСТИРОВАНИЕ ПРОГРАММЫ 15](#_Toc59474494)

[ЗАКЛЮЧЕНИЕ 21](#_Toc59474495)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc59474496)

[ПРИЛОЖЕНИЕ А 23](#_Toc59474497)

[ПРИЛОЖЕНИЕ Б 25](#_Toc59474498)

[ПРИЛОЖЕНИЕ В 28](#_Toc59474499)

# Введение

Стремительное развитие и широкое распространение вычислительной техники послужили предпосылками к появлению нового раздела науки, названного информатикой. Сегодня информатика не только научная и учебная дисциплина. Можно с полным основанием утверждать, что она превратилась в динамично развивающуюся отрасль народного хозяйства. Всё больше расширяется сфера применения компьютеров. ЭВМ применяются теперь не только в таких традиционных областях как кибернетика, финансы, экономика. Сегодня компьютеры приходят даже в те сферы, где главную роль всегда играл человек: здравоохранение, культура, образование[1].

Данный курсовой проект предусматривает написание простейшего windows-приложения, используя набор базовых функций winapi.

**Windows API** (англ. *application programming interfaces*) — общее наименование целого набора базовых функций интерфейсов программирования приложений операционных систем семейств Windows и Windows NT корпорации «Майкрософт». Является самым прямым способом взаимодействия приложений с Windows. Для создания программ, использующих Windows API, «Майкрософт» выпускает SDK, который называется Platform SDK и содержит документацию, набор библиотек, утилит и других инструментальных средств.

Windows API был изначально спроектирован для использования в программах, написанных на языке C (или C++). Работа через Windows API — это наиболее близкий к системе способ взаимодействия с ней из прикладных программ. Более низкий уровень доступа, необходимый только для драйверов устройств, в текущих версиях Windows предоставляется через Windows Driver Model.

Версии:

* **Win16** — первая версия Windows API для 16-разрядных версий Windows. Изначально назывался просто Windows API, затем стал называться Win16 для отличия от Win32.
* **Win32s** — подмножество Win32, устанавливаемое на семейство 16-разрядных систем Windows 3.x и реализующее ограниченный набор функций Win32 API для этих систем.
* **Win32** — 32-разрядный API для современных версий Windows. Самая популярная ныне версия. Базовые функции этого API реализованы в DLL **kernel32.dll** и **advapi32.dll**; базовые модули GUI — в **user32.dll** и **gdi32.dll**. Win32 появился вместе с Windows NT и затем был перенесён (в несколько ограниченном виде) в системы серии Windows 9x. В современных версиях Windows, происходящих от Windows NT, работу Win32 GUI обеспечивают два модуля: csrss.exe (*Client/Server Runtime Subsystem*), работающий в пользовательском режиме, и win32k.sys в режиме ядра. Работу же системных Win32 API обеспечивает ядро — ntoskrnl.exe.
* **Win64** — 64-разрядная версия Win32, содержащая дополнительные функции для использования на 64-разрядных компьютерах. Win64 API можно найти только в 64-разрядных версиях Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2 и Windows 7[2].

# 1 Анализ задания и постановка задач

С помощью анализа задач определяют, что пользователю нужно сделать в плане действий для выполнения той или иной задачи. Детальный анализ задач используется для понимания существующей системы и того, как в ней движется информация. Эти знания важны как для поддержания существующей системы, так и для правильного построения новой. С помощью этого метода можно точно определить какие функции должны быть включены в систему, и как должен выглядеть интерфейс этих функций[3].

В нашем случае предметной областью является мини-игра «LINES». Игра существует уже давно, поэтому ничего нового изобретать нам не придётся, нужно лишь реализовать приложение.

Правила игры предельно просты:

* игровое поле представляет собой квадратную сетку ячеек, в которых располагаются шары, но не более одного в каждой из них;
* на каждом ходу в выбранных случайным образом ячейках появляются новые шары;
* игрок с помощью «мыши» должен переместить один из шаров в свободную ячейку, стараясь выстраивать из одинаковых шаров непрерывные горизонталные, вертикальные или диагональные линии;
* если после перемещения или появления шара линия достигнет заданной длины, то все шары, входящие в нее, удаляются. При этом игрок набирает определнное количество очков;
* перемещение шара из исходной ячейки в «ячейку назначения» допускается, если существует путь, проходящий по свободным ячейкам, расположенным рядом друг с другом по вертикали или по горизонтали;
* если после перемещения шара была удалена линия, то на следующем ходу новые шары, которые должны были бы появиться в соответствии с п.2, не появляются;
* игра заканчивается, когда на игровом поле не остается свободных ячеек – они все заполнены шарами;
* цель игры – набрать как можно больше очков, которые запоминаются в соответствующей таблице;
* в игре используется «шкала ценностей». При этом количество очков, которые получает игрок при составлени линии шаров, вычисляется (при *n ≥ k*) по формуле *q = n ∙ (n – k + 1)* (*n* – количество выстроенных в линию шаров, *k* – минимальное количество удаляемых шаров, *q* – получаемые очки);
* в любой момент игры на экран выводится подсказка о шарах, которые появятся на следующем шаге. Эти подсказки отображаются на поле в виде уменьшенных изображений шаров.

Что касается дополнительного функционала, то в прогрмме должна быть возможность изменять:

* размеры игрового поля;
* минимальное количетво шаров, которые удаляются при выстраивании в линию;
* количество появляющихся шаров на каждом шаге.

Следовательно, в нашей системе четко прослеживается необходимость в реализации следующих функций:

* перемещение шара;
* генерация новых шаров;
* подсчет очков;
* изменения размера поля;
* таблица рекордов;
* звуковое сопровождение;
* подсказка о генерации следующих шаров;
* проверка на заполеннность поля.

# 2 Проектирование программы

Проектирование — процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект — целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы. Наряду с анализом требований, является частью большой стадии жизненного цикла системы, называемой определением системы. Результаты этой стадии являются входной информацией для стадии реализации системы[4].

Главный этап на стадии проектирования программы – это определение ее ключевых составных частей, каждая из которых несет ответственность за определенные задачи, решаемые в процессе выполнения программы, и является относительно автономной либо частично зависимой от других частей[5].

Для выполнения всех необходимых условия, поставленных при анализе и поставке задачи необходимо реализовать два главных модуля: автомат «Управления игрой» и автомат «Ячейка».

Автомат «Управления игрой» обеспечивает управление:

* передвижением и прыжками шаров;
* появлением и удалением шаров;
* поиском кратчайшего пути.

Общая схема автомат и его функциональных переходов представлена в приложении.

Автомат «Ячейка» обеспечивает:

* установку шара в ячейке;
* удаление шара;
* подсказку о том, что шар на следующем ходу должен появиться;
* отображения состояния ячейки на экране.

Общая схема автомат и его функциональных переходов представлена в приложении.

# 3 Реализация программы

После проектирования программы следует реализация программы. На данному этапе важно придерживаться описанного ранее плана и принципов разработки программы. Важно помнить, что разрабатываемый продукт в любой момент времени может быть модифицирован без затруднений, поэтому код должен быть читабелен и содержать документацию для каждого разработанного модуля и функции[6].

В игровом приложении в соответствии с разделом проектирования программы были разработаны следующие функции:

* *int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) –* главная функция окна . Выполняет вызов всех необходимых функция для инициализации и обраотки сообщений.
* *ATOM MyRegisterClass(HINSTANCE hInstance)* – регистрация класса окна.
* *BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)* – создание и отображения окна приложения. Функция *InitInstance* представлена в листинге 3.1.

**Листинг 3.1** – Функция *InitInstance*

1. BOOL InitInstance(HINSTANCE hInstance, int nCmdShow) {
2. hWnd = CreateWindow(
3. szWindowClass,
4. szTitle,
5. WS\_CAPTION | WS\_OVERLAPPED | WS\_MINIMIZEBOX | WS\_SYSMENU,
6. CW\_USEDEFAULT, 0, 1600, 600, NULL, NULL, hInstance, NULL);
7. if (!hWnd) {
8. return FALSE;
9. }
10. ShowWindow(hWnd, nCmdShow);
11. UpdateWindow(hWnd);
12. return TRUE;
13. }

* *LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)* – главная функция обработки сообщения окна приложения. Функция обрабатывает такие сообщения как: создение окна, нажатие мыши, обработка сообщений от таймера, вызов дочерних окон, уничтожение окна.
* *LRESULT CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)* – главная функция окна диалога с информацией о приложении: как называется и кто разработал. Функция *About* представлена в листинге 3.2.

**Листинг 3.2** – Функция *About*

1. BOOL InitInstance(HINSTANCE hInstance, int nCmdShow) {
2. hWnd = CreateWindow(
3. szWindowClass,
4. szTitle,
5. WS\_CAPTION | WS\_OVERLAPPED | WS\_MINIMIZEBOX | WS\_SYSMENU,
6. CW\_USEDEFAULT, 0, 1600, 600, NULL, NULL, hInstance, NULL);
7. if (!hWnd) {
8. return FALSE;
9. }
10. ShowWindow(hWnd, nCmdShow);
11. UpdateWindow(hWnd);
12. return TRUE;
13. }

* *LRESULT CALLBACK Custom(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) –* главная функция окна диалога, которая отвечает за получения данных от пользователя для настроки игрового поля: размер, количество шаров для удаления и количество шаров для генерации.
* *LRESULT CALLBACK BestResults(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) –* главная функция окна дилога с таблицей лидеров. Осуществляет вывод данных с информацией о рекордах.
* *LRESULT CALLBACK GetName(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) –* главная функция окна диалога получения имени игрока. Являетя обработчиком для данного окна.
* *void Alines(int e) –* автомат «Управления игрой». Данная выполняет всю обработку сообщения при взаимодействии пользователя с игрой. В зависимости от различных действий, вызывает различные функцию. Обработка приходится на следующие операции: поиск игрового шара, передвижение игрового шара, удаление линии, появление новых шаров и анимация шара.
* *bool xk0() –* проверка на пустоту ячейки.
* *bool xk1() –* проверка на наличие шара в ячейке.
* *bool xk2() –* проверка на наличие прыгающего шара.
* *bool x0() –* проверка на наличие пути от активной ячейки до выбранно.
* *bool x1() –* проверка на наличе ячейки, в которую шар еще должен перейти для того, чтобы попасть в выбранную ячейку.
* *bool x2() –* проверка на наличие линий для удаления.
* *bool x3() –* проверка на окончание удаления.
* *bool x4() –* проверка на окончание появления шаров.
* *bool x5() –* удаление линий после появления шаров. Функция *x5* представлена в листинге 3.3.

**Листинг 3.3** – Функция *x5*

1. bool x5() {
2. itr = appear\_list.begin();
3. while (itr != appear\_list.end()) {
4. CheckLines(\*itr++);
5. }
6. return (explode\_list.size() != 0);
7. }

* *bool z0() –* запуск анимации прыжка шара.
* *bool z1\_1() –* запуск анимации передвижения шара.
* *bool z1\_2() –* окончаие анимации прыжка.
* *bool z1\_3() –* осуществление анимации прыжка.
* *bool z2\_1() –* передвижения шара на следующую ячейку.
* *bool z2\_2() –* окончание передвижения. Функция *z2\_2* представлена в листинге 3.4.

**Листинг 3.4** – Функция *z2\_2*

1. void z2\_2() {
2. KillTimer(hWnd, 1);
3. }

* *bool z2\_3() –* запуск удаления шаров.
* *bool z2\_4() –* запуск появления шаров.
* *bool z3\_1() –* удаление шаров. Осуществляет удаление ранее тех шаров, которые были помечены, как удаляемые.
* *bool z3\_2() –* обновляет табло. Осуществляет подсчет удаленных шаров и вызвает функцию отвечающую за перерисвоку верхней части приложения. Функция *z3\_2* представлена в листинге 3.5.

**Листинг 3.5** – Функция *z3\_2*

1. void z3\_2() {
2. KillTimer(hWnd, 1);
3. gamescore += (explode\_list.size() - del\_balls + 1) \* explode\_list.size();
4. DrawScore();
5. explode\_list.clear();
6. }

* *bool z4\_1() –* появление шаров на игровом поле. Осуществляет заполнение ранее сгенерированных ячеек соответствующими шарами.
* *void z4\_2()* – осуществляет вызов функции *GenerateAppearList()*. Данная функция вызывается в момент появления шаров на игровом поле.
* *bool checkLines(const cell& in)* – проверяет проходит ли через ячейки линия, которая подлжеит удалению. Выполняет проверку состояния для каждой клетки с параллельным занесением во временный массив. Если удалось набрать необходимое количество шаров для удаления, тогда временный массив сохраняется.
* *void CheckAppearList()* – проверка списка появляющихся шаров на предмет занятости соответствующей ячейки. Функция *CheckAppearList* представлена в листинге 3.6.

**Листинг 3.6** – Функция *CheckAppearList*

1. void CheckAppearList() {
2. int tmp;
3. itr = appear\_list.begin();
4. while (itr != appear\_list.end()) {
5. if (itr->State() == 3) {
6. tmp = itr->PreColor();
7. itr->PreColor() = -1;
8. FindEmptyCell(\*itr);
9. itr->PreColor() = tmp;
10. (\*itr).ACell(2);
11. }
12. ++itr;
13. }
14. }

* *void GenerateAppearList()* – создание списка появляющихся шаров. Удаляет информацию о прошлых шарах, поиск пустых ячеек и генерация шаров.
* *bool FindEmptyCell(cell& in)* – поиск пустой ячейки. Функция-предикат определяет является ли ячейка пустой. Функция *FindEpmtyCell* представлена в листинге 3.7.

**Листинг 3.7** – Функция *FindEmptyCell*

1. bool FindEmptyCell(cell& in) {
2. cell l;
3. l.posx = random(max\_x);
4. l.posy = random(max\_y);
5. if (l.State() == 0) {
6. in = l;
7. return true;
8. };
9. for (int i = 0; i < max\_x \* max\_y; i++) {
10. if (l.posx != max\_x - 1) {
11. l.posx++;
12. }
13. else if (l.posy != max\_y - 1) {
14. l.posy++;
15. l.posx = 0;
16. }
17. else {
18. l.posx = 0;
19. l.posy = 0;
20. };
21. if (l.State() == 0) {
22. in = l;

Продолжение листинга 3.7

1. return true;
2. }
3. }
4. GameOver();
5. return false;
6. }

* *bool Valid(const cell &in)* – проверяет можно ли через передаваемую ячейку пройти. Через ячейку можно пройти в том случае, если она не выходит за рамки игрового поля и она не занята другим шаром.
* *void NewGame()* – подготовка ресурсов для новой игры. Функция выполняет построение поля и его заполенние, обновление счетчика времени и очков. Функция *NewGame* представлена в листинге 3.8.

**Листинг 3.8** – Функция *NewGame*

1. void NewGame()
2. {
3. for (int i = 0; i < max\_x; i++) {
4. for (int j = 0; j < max\_y; j++) {
5. map[i][j].y = 0;
6. map[i][j].color = 0;
7. map[i][j].pre\_color = -1;
8. map[i][j].num\_pic = 0;
9. }
10. }
11. y\_lines = 0;
12. gamescore = 0;
13. gametime = 0;
14. SetTimer(hWnd, 0, 1000, NULL);
15. cell l;
16. for (int i = 0; i < del\_balls; i++) {
17. FindEmptyCell(l);
18. ball\_color = random(7);
19. l.ACell(1);
20. }
21. GenerateAppearList();
22. }

* *void GameOver()* – обработка окончания игрового процесса. Если режим игры не был настроен вручную и пользователю удалось побить рекорд для соответствующего режима, тогда выполняется отрисовка окна, запрашивающего имя игрока, иначе выполняется отрисовка окна с таблицей лидеров. Если же пользователь настраивал режим игры вручную, тогда выполняется отрисовка окна с информаций о проигрыше. Затем вызываеться функция, которая выполни старт новой игры.
* *void DrawTime()* – отрисовка на табло продолжительности игры. Сначала функция осуществляет вычисление продолжительности игры, а затем выводит полученное значение в формате чч:мм:сс на табло.
* *void DrawScore()* – отрисовка текущих очков. Функция осуществляет вывод очков, набранных пользователем. Функция *DrawScore* представлена в листинге 3.9.

**Листинг 3.9** – Функция *DrawScore*

1. BOOL InitInstance(HINSTANCE hInstance, int nCmdShow) {
2. hWnd = CreateWindow(
3. szWindowClass,
4. szTitle,
5. WS\_CAPTION | WS\_OVERLAPPED | WS\_MINIMIZEBOX | WS\_SYSMENU,
6. CW\_USEDEFAULT, 0, 1600, 600, NULL, NULL, hInstance, NULL);
7. if (!hWnd) {
8. return FALSE;
9. }
10. ShowWindow(hWnd, nCmdShow);
11. UpdateWindow(hWnd);
12. return TRUE;
13. }

* *void DrawTop()* – отрисовать табло. Функция осуществляет окраску верхней части приложения и вызывает функци для отрисовки набранных очков и времени.
* *void CheckCustomParameters()* – проверить заданные параметры игры. Данная функция осуществляет проверку на корректность настроек игры. Если игра настроена не верно, функция установит стандартное поведения для игры.
* *void GetInfo()* – получение информации о игре. Функция открывает файл для чтения и если получилось открыть, тогда считывает информацию о режиме игры, количестве удаляемых шаров, количестве шаров для генерации, размеры поля и таблицу лидеров. Если открыть файл не удалось, то устанавливает стандартные настройки игры и заполняет таблицу лидеров игроками с минимальным порогом.
* *void WriteInfo()* – сохранение информации о игре. Функция открывает файл для записи. Если файл получилось открыть, тогда выполняется запись текущего режима игры, размера поля, количество шаров для удаления, количество генерируемых шаров и таблицу лидеров. Данная функция нужна для того, чтобы при следующем открытии приложения можно было восстановить всю информацию о игре. Функция *WriteInfo* представлена в листинге 3.10.

**Листинг 3.10** – Функция *WriteInfo*

1. void WriteInfo() {
2. FILE\* out;
3. if (out = fopen("leaders.dat", "wb")) {
4. fwrite(&gametype, sizeof(int), 1, out);
5. fwrite(&max\_x, sizeof(int), 1, out);
6. fwrite(&max\_y, sizeof(int), 1, out);
7. fwrite(&app\_balls, sizeof(int), 1, out);

Продолжение листинга 3.10

1. fwrite(&del\_balls, sizeof(int), 1, out);
2. for (int i = 0; i < 3; i++) {
3. fwrite(&leaders[i], sizeof(info), 1, out);
4. }
5. fclose(out);
6. }
7. }

# 4 Тестирование программы

Тестирование программы – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Тестирование программы и результаты представлены в таблице 4.1

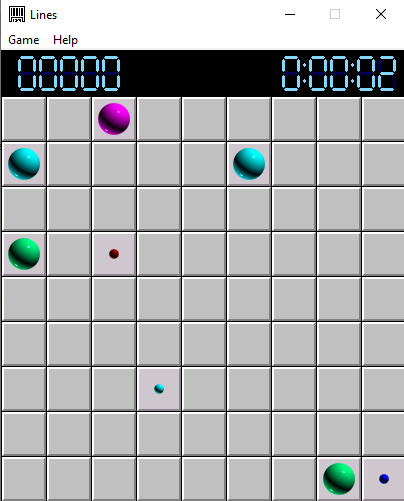
**Таблица 4.1** – Тестирование программы

|  |  |  |
| --- | --- | --- |
| **Действие** | **Ожидаемый**  **Результат** | **Полученный результат** |
| Запуск программы | Перед пользователем открывается игровое поле с обнуленным счетом и расставленными шарами. | Игровое поле сгенерировано. |
| Перемещение шарика | После выбора шара и места, куда для перемещения, мяч перемещяется. | Местоположение шара изменяется. |
| Попытка переместить шарик в закрытую область | Шар перемещается только в доступные для перемещения места. | Шар не перемещается в места, которые закрыты для его прохода. |
| Новая игра | Обновление всех данных поля: время, счет, шары. | Выполняется обновление игрового поля. |
| Выход | Завершения программы и освобождения памяти. | Закрытие и освобождение памяти. |
| Составление шаров в ряд | После того, как в ряд будет собранно заданное количество шаров, полоса должна исчезнуть и добавить очки пользователю. | Шары удаляются, очки добавляются. |
| Подсчет очков | Подсчет очков осуществляется только после удаления шаров и по определенной формуле. | Очки добавляются в соответствии с формулой. |
| Настройка игры | Пользователь имеет возможность изменить размеры поля, количество генерируемых шаров и количество шаров необходимы для удаления. | Настройки изменяют поведение программы. |

Окончание таблицы 4.1

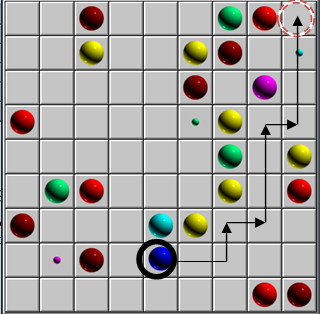
|  |  |  |
| --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Полученный результат** |
| Работа меню приложения | Вывод меню осуществляется по нажатию. В меню можно выбрать соответствующие пункты: новая игра, настройки, сложность режима, выход и информация о приложении. | Осуществляется вывод меню и возможность взаимодействия с ним. |
| Фильтрация | Вывод небольшого окна с информацией о приложении. | Пользователю открывается откно с информацией о приложении. |
| Таблица рекордов | Если пользователь проигрывает и на момент проигрыша у пользователя большее количество очков, чем у игрока в таблице, должно быть предложено ввести его имя для занесения в таблицу. | После окончания игры пользователю предлагается ввести имя для внесения в таблицу. |

Сразу после запуска игрового приложения, выполняется генерация поля, очков, времени и добавления шаров на поле. Пользователь сразу может приступить к игре используя компьютерную мышь. Игровое поле представлено на рисунке 4.1.



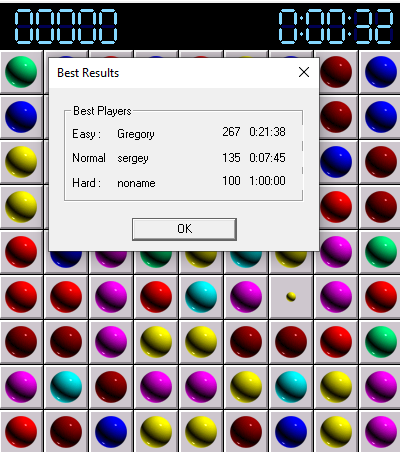
**Рисунок 4.1** – Запуск приложения

Когда пользователь нажимает на шарик, а затем на пустую клетку, выполняется поиск кратчайшего пути к этой клетке. Если таков путь имеется, тогда выполняется перемещение. Принцип работы перемещения шара представлени на рисунке 4.2.



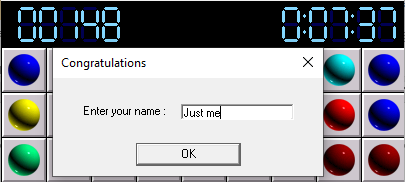
**Рисунок 4.2** – Перемещение шара

Игра будет продолжать до тех пор, пока есть свободные ячейки для перемещения. Если на момент окончания игры, набранных очков не хватило для внесения в таблицу лидеров, выполняется отрисовка окна с информацией о лидерах по всем режимам. Таблица лидеров представлена на рисунке 4.3.



**Рисунок 4.3** – вывод таблицы лидеров в случае недобора очков

Если же пользователь в ходе игры сумел набрать необходимое количество очков для попадания в таблицу лидеров, тогда выполняется отрисовка окна, которое запршаивает имя игрока. Окно, запрашивающее имя игрока, предсталвено на рисунке 4.4.



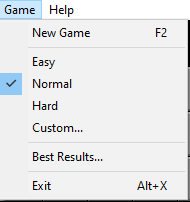
**Рисунок 4.4** – запрос имя игрока в случае добора очков

Таблицы рекордов для игры настроенной вручную не предусмотрено, поэтому в случае проигрыша, отрисовывается окно, которое сообщает о проигрыше даже в случае набора большого количества очков. Информация о проигрыше представлена на рисунке 4.5.



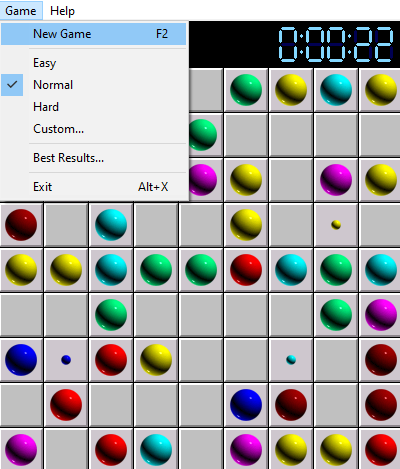
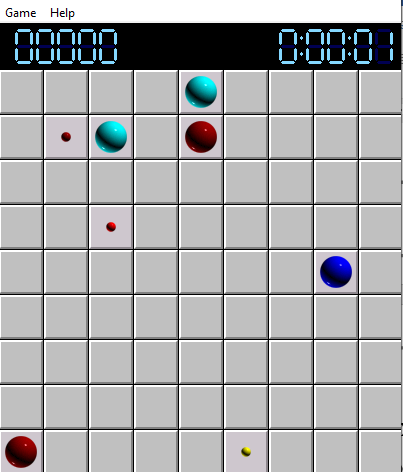
**Рисунок 4.5** – информация о проигрыше в режиме ручной настройки

На навигационной панели располагается две кнопки: «**Game**» и «**About**». По нажатию на «**Game**» выводится меню, которое содержит такие пункты, как: начало новой игры, изменения уровня сложности, таблица лидеров и выход из приложения. Меня «**Game**» предсталвено на рисунке 4.6.



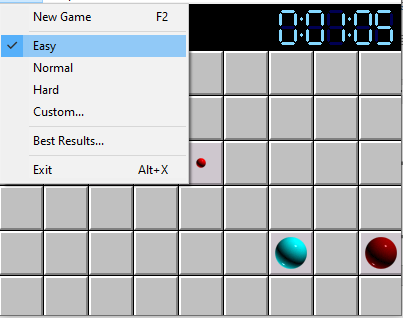
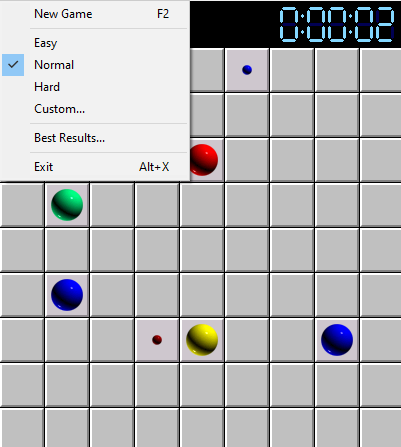
**Рисунок 4.6** – меню «**Game**»

Пункт меню, отвечающий за начало новой игры, очищает игровое поле, обнуляет время и счетчик очков, а затем генерирует новые шары. Пример работы пункта меню, отвечающего за начало новой игры, предсталвен на рисунке 4.7.

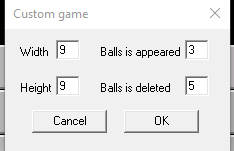
**Рисунок 4.7** – До и после нажатия на «**New game**»

Далее в списке меню идут пункты отвечающие за сложность игры (размер поля, количество шаров для удаления, количество шаров для генерации). Пример использования пунктов, отвечающих за изменения режима игры, представлен на рисунке 4.8.

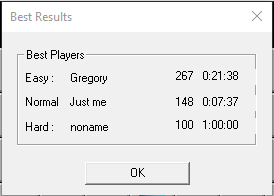
**Рисунок 4.8** – выбор режима игры

Также есть возможность настроить сложность игры вручную. Пример окна, отвечающего за ручную настройку приложения представлен на рисунке 4.9.



**Рисунок 4.9** – настройка сложности игры вручную

У пользователя в любой момент времени есть возможность просмотреть таблицу лидеров. Таблица лидеров представлена на рисунке 4.10.



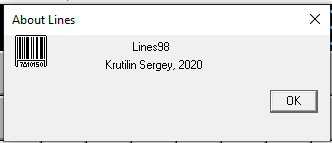
**Рисунок 4.10** –таблица лидеров

По нажатию на «**About**» выводится другое меню, предлагающее пользователю вывести информацию о приложении. Меню «**Help»** представлено на рисунке 4.11.



**Рисунок 4.11** – меню «**Help**»

При нажатии на пункт меню в данном случае выполняется отрисовка окна с информацией о приложении. Окно с информацией о приложении представлено на рисунке 4.12.



**Рисунок 4.12** – информация о приложении

# Заключение

Одной из задач курсовой работы было умение применять полученные знания на практике. В данной курсовой работе были применены все знания, полученные в течение обучения предмету. Для данной задачи было разработано игровое приложение «Линии», проект которой реализован в среде разработки *Visual Studio 2019*. При решении поставленных задач, а именно: показать знания и рабочее приложение – требовалось реализовать следующие функции: отрисовка окна, поиск пути для перемещения, анимация перемещения, выбор режима игры, таблица лидеров. Помимо этого требовалось реализовать в приложении дружественный интуитивно понятный интерфейс. Вместе с этим надо было проверить ввод данных, его корректность и правильность.

Программа написана модулями, работает без сбоев и ошибок, все модули подключены и исполняют свои роли. Также в программе реализованы все необходимые для проекта функции, которые были заданы в курсовой работе.

При тестировании программы была доказана его работоспособность и возможность использования пользователями.

Таким образом программа реализована в соответствии всем требованиям и стандартам, тщательно протестирована и проблем с работой не выявлено. Таким образом работа может использоваться для решения поставленных при создании базы данных задач. Задачи, связанные с реализацией кода и его тестированием выполнены успешно.

# Список использованных источников

1. Эллайн Аллекс. С++. От ламера до программера./ Эллайн Аллекс – Питер, 2015. – 480 с;
2. Брайан Керниган. Практика программирования./ Брайан Кернига, Роб Пайк – [Вильямс](https://oz.by/producer/more1207.html), 2004. – 288 c;
3. Культин, Н. С/С++ в задачах и примерах / Н. Культин. – СПб. : БХВ-Петербург, 2007. – 288 c.
4. Фридман, А. С/С++. Алгоритмы и приемы программирования / А. Фридман. – М. : Бином, 2007. – 560 с.
5. Липпман Б. Язык программирования C++. Базовый курс/ Б. Липпман, 2001 – 1104с.
6. Павловская Т.А. С++ Программирование на языке высокого уровня. Учебник для вузов/ Павловская Т.А – Москва, 2003 – 450с.

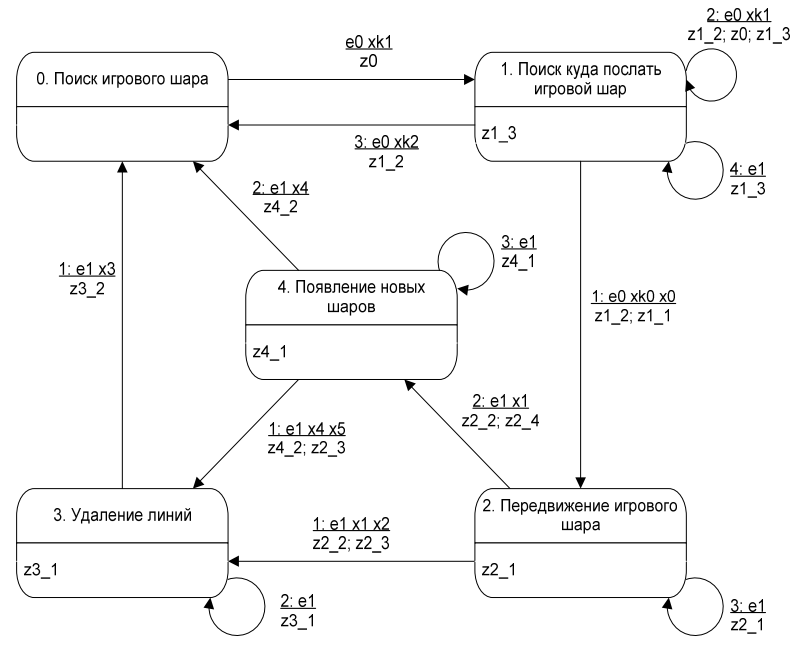
# ПРИЛОЖЕНИЕ А

(обязательное)

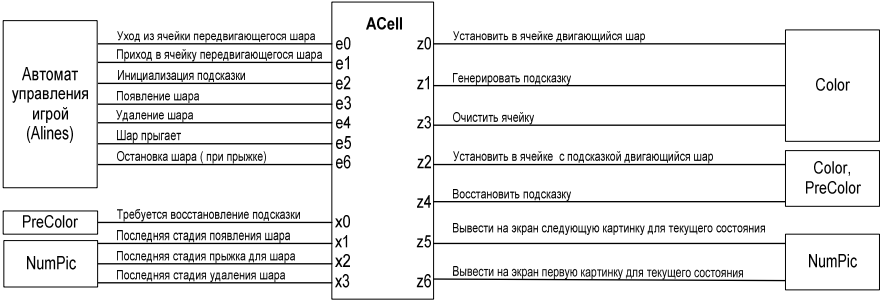
**Блок схема модульной системы приложения**



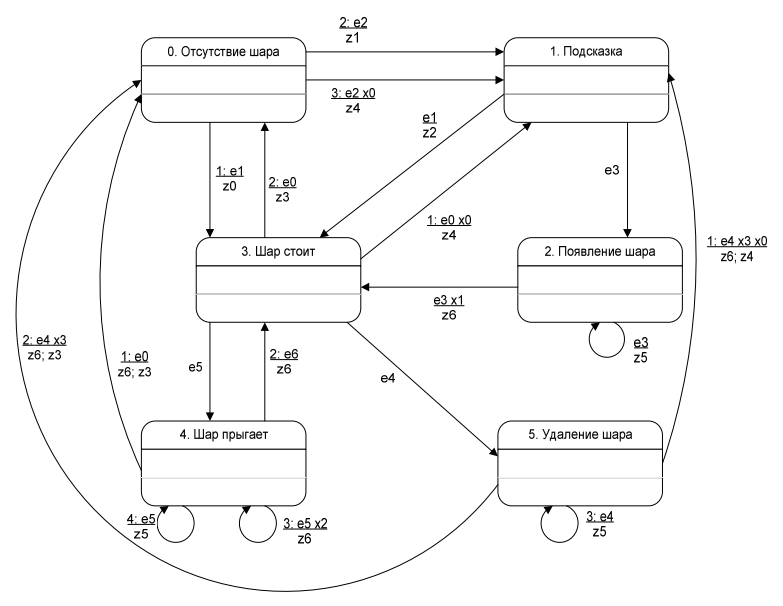
**Рисунок А.1** – Схема связей автомата «Управления игрой»



**Рисунок А.2** – Граф переходов автомата «Управления игрой»



**Рисунок А.3** – Схема связей автомата «Ячейка»

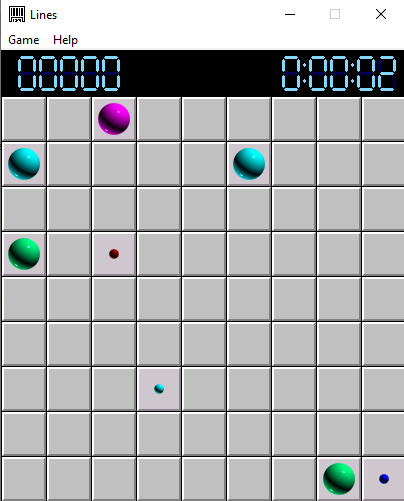


**Рисунок А.4** – Граф переходов автомата «Ячейка»

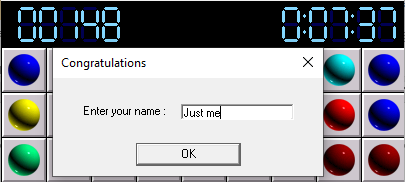
# ПРИЛОЖЕНИЕ Б

(обязательное)

**Интерфейс игрового приложения**



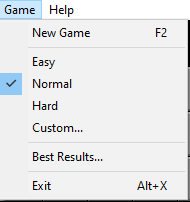
**Рисунок Б.1** – игровое приложение



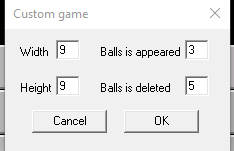
**Рисунок Б.2** – запрос «никнейма» игрока



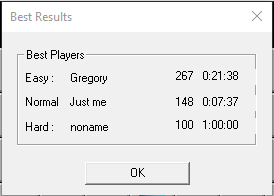
**Рисунок Б.3** – информация о проигрыше в режиме ручной настройки



**Рисунок Б.4** – меню «**Game**»



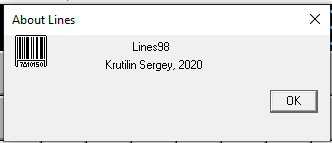
**Рисунок Б.5** – настройка сложности игры вручную



**Рисунок Б.6** – таблица лидеров



**Рисунок Б.7** – меню «**Help**»



**Рисунок Б.8**  – информация о приложении

# ПРИЛОЖЕНИЕ В

(обязательное)

**Листинг программы (в электронном виде).**

Листинг игрового приложения представлен на диске в файле Листинг.docx