Министерство науки и высшего образования РФ
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Разработка программы сложной структуры методом нисходящего программирования. Игра “Сапёр”»

Выполнил: ст. гр. 25ВВВ4
Исянов Д.Ф.
Принял:
Кандидат т.н. Токарев А. Н.

Пенза 2026

Министерство науки и высшего образования РФ
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по курсу «Программирование»
на тему «Разработка программы сложной структуры методом нисходящего программирования. Игра “Сапёр”»

Выполнил: ст. гр. 25ВВВ4
Исянов Д.Ф.
Принял:
Кандидат т.н. Токарев А. Н.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

_____ М.А. Митрохин

«____»_____20__г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Исенову Ринару Фяритовичу Группа 25ВВВ4

Тема проекта: ***Разработка программы сложной структуры методом нисходящего программирования.***

Исходные данные (технические требования) на проектирование

Игра сапёр. Требования:
- система сохранения результатов игр
- система набора очков
- разные уровни сложности
- управление осуществляется с помощью мыши

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. **Расчетная часть**

Разработка программы.

2. **Графическая часть**

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. **Экспериментальная часть**

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "18" 02 2026

Дата защиты проекта "____" _____

Руководитель _____

Задание получил "26" 02 2026г.

Студент Исянов Ринар Фердинович

Содержание

Введение 8

Постановка задачи 10

Выбор решения 12

Описание разработки программы 15

Отладка и тестирование 19

Описание программы 21

Руководство пользователя 26

Заключение 30

Список используемых источников 32

Приложение А. Листинги программы ..... **Error! Bookmark not defined.**

common.h ..... **Error! Bookmark not defined.**

engine.h ..... **Error! Bookmark not defined.**

records.h ..... **Error! Bookmark not defined.**

ui.h ..... **Error! Bookmark not defined.**

engine.c ..... **Error! Bookmark not defined.**

records.c ..... **Error! Bookmark not defined.**

ui.c ..... **Error! Bookmark not defined.**

main.c ..... **Error! Bookmark not defined.**

Приложение В. Результаты работы программы 33

Введение

Компьютерные игры являются одним из наиболее наглядных примеров программного обеспечения, в котором одновременно сочетаются алгоритмы сложной обработки данных, работа с интерактивным пользовательским вводом, динамический вывод информации на экран, долгосрочное хранение результатов и строгая организация исходного кода по модулям. Даже классическая игровая логика требует от разработчика проектирования продуманной структуры: необходимо эффективно хранить состояние игрового поля, обрабатывать координатные события ввода, реализовывать рекурсивные алгоритмы обхода элементов, проверять условия победы или поражения и своевременно обновлять интерфейс.

В данной курсовой работе рассматривается разработка консольного приложения, реализующего логику классической игры «Сапёр». Пользователь взаимодействует с игровым полем в режиме реального времени, осуществляет поиск скрытых мин, выставляет маркеры (флаги) на опасные ячейки и открывает безопасные зоны. В программе предусмотрены три настраиваемых уровня сложности (легкий, средний и сложный), масштабирующих размер игрового поля и количество скрытых угрозы, а также полноценная система сохранения рекордов в конфигурационный файл.

Разрабатываемая программа предназначена не только для наглядной демонстрации игрового процесса, но и для закрепления практических навыков проектирования сложной архитектуры программного обеспечения, работы с многомерными массивами данных, текстовыми файлами, построения консольных интерфейсов и низкоуровневой обработки событий пользователя.

Особенностью архитектуры создаваемого приложения является использование многомодульной структуры на стыке языков высокого и низкого уровней (C и ASM). Разделение монолитного кода на заголовочные

файлы и файлы реализации (main.c, engine.c, ui.c, records.c, common.h) позволило изолировать игровой цикл, логику генерации мин, отрисовку интерфейса и подсистему работы с файлами рекордов. Такой подход строго соответствует методу нисходящего проектирования: от определения общей концепции и высокоуровневой логики главного меню к декомпозиции на конкретные подзадачи, реализуемые обособленными модулями.

Приложение функционирует в среде консоли ОС Windows и активно использует возможности интерфейса WinAPI для управления положением курсора, принудительного программного отключения режимов быстрого редактирования консоли и перехвата системных очередей ввода. Ключевой технической деталью является обработка событий компьютерной мыши внутри консольного буфера, что обеспечивает интуитивно понятный для пользователя игровой процесс. Часть логики, отвечающая за опрос состояния управляющих клавиш (в частности, функция организации паузы), реализована с применением ассемблерных вставок (__asm), что демонстрирует эффективную интеграцию низкоуровневых команд процессора архитектуры x86 в общую структуру программы на языке С.

Постановка задачи

Необходимо разработать консольное приложение, реализующее классическую игру «Сапёр» для операционной системы Windows. Программа должна иметь интуитивно понятный текстовый интерфейс, интерактивное главное меню, выбор уровня сложности, динамически генерируемое игровое поле, а также функции отображения текущего игрового прогресса (количества открытых ячеек) и сохранения результатов.

Основная цель игры заключается в полном очищении игрового поля от безопасных ячеек без детонации скрытых мин. Пользователь должен иметь возможность открывать ячейки и помечать предполагаемые места расположения мин специальными маркерами (флагами). Игра завершается поражением при клике на мину или победой при успешном открытии всех пустых клеток. В зависимости от выбранного уровня сложности (легкий, средний, сложный) должны изменяться размеры двумерной сетки поля и пропорционально масштабироваться количество размещаемых мин.

В ходе разработки необходимо определить структуры данных для эффективного хранения состояния игрового пространства: координат ячеек, признаков их скрытости/открытости, наличия мин и установленных флагов, а также записей таблицы результатов. В качестве базовых структур данных необходимо использовать многомерные (двумерные) массивы фиксированного размера, глобальные переменные состояния игрового процесса и систему указателей.

Программа должна быть спроектирована с использованием многомодульной архитектуры, разделяющей логику на заголовочные файлы и файлы реализации. В соответствии со структурой проекта необходимо выделить следующие компоненты:

— main.c — содержит точку входа в программу, инициализацию системных дескрипторов консоли и главный цикл обработки меню.

— engine.c / engine.h — реализует ядро игрового процесса: инициализацию параметров сложности, случайную генерацию мин на поле, подсчет смежных угроз и рекурсивный алгоритм автоматического открытия пустых зон.

— ui.c / ui.h — отвечает за функции позиционирования курсора, принудительную очистку экрана, низкоуровневый опрос устройств и поклеточнуюотрисовку графического интерфейса поля.

— records.c / records.h — организует файловый ввод-вывод для чтения, форматированного вывода и добавления новых записей в таблицу рекордов.

— common.h — обеспечивает сквозное объявление глобальных переменных состояния игры для исключения дублирования памяти при линковке модулей.

Основным устройством ввода является компьютерная мышь, события которой перехватываются внутри консольного буфера: нажатие левой кнопки мыши инициирует открытие ячейки, нажатие правой кнопки — установку или снятие флага. Дополнительным устройством ввода выступает клавиатура (использование клавиши P для перевода текущей сессии в режим паузы и возврата в меню). Вывод информации осуществляется на экран монитора в символьном режиме Windows-консоли с кодировкой CP1251.

После успешного завершения игры программа должна запрашивать имя пользователя и осуществлять дозапись итогового счета в текстовый файл рекордов. Таблица лидеров должна динамически считываться из файла по требованию пользователя через главное меню приложения.

В результате выполнения задачи должно быть создано стабильное, отказоустойчивое консольное приложение с развитым игровым процессом, управлением мышью, модульной структурой кода и подсистемой хранения результатов.

Выбор решения

Для реализации проекта выбран язык программирования C, так как он обеспечивает высокую производительность, предоставляет прямой доступ к оперативной памяти и низкоуровневым структурам данных, а также позволяет нативно взаимодействовать с функциями операционной системы через интерфейсы прикладного программирования (API). В качестве среды разработки применяется интегрированная среда MicrosoftVisualStudio, проект скомпилирован как консольное приложение Windows.

Игровое пространство представляет собой прямоугольную сетку (двумерную матрицу) переменного размера, ограниченную константой MAX = 30. В зависимости от выбранного пользователем уровня сложности, параметры поля динамически масштабируются:

— Легкий уровень: сетка 9*9 ячеек, 10 скрытых мин.

— Средний уровень: сетка 16*16 ячеек, 40 скрытых мин.

— Сложный уровень: сетка 30*16 ячеек, 99 скрытых мин.

Для визуализации элементов поля в текстовом консольном режиме используются следующие символьные маркеры:

— Неисследованная (закрытая) ячейка — символ точки.

— Установленный игроком маркер мины (флаг) — символ F

— Пустая открытая ячейка, не имеющая мин в смежных позициях — пробел

— Открытая ячейка, граничащая с минами — цифра от 1 до 8, указывающая на количество угроз в радиусе одной клетки.

— Скрытая мина (отображается только в момент детонации при проигрыше) — символ звездочки *

При запуске приложения выполняется базовая инициализация окружения: настройка кодовой страницы CP1251 для корректного отображения кириллических символов, инициализация генератора

псевдослучайных чисел srand() на основе системного времени для уникального распределения мин в каждой сессии, а также получение дескриптора стандартного ввода STD_INPUT_HANDLE.

Главное меню программы является интерактивным и содержит следующие пункты: «Возобновить игру» (динамически появляется, если сессия активна и находится на паузе), «Начать новую игру» (с последующим переходом в подменю выбора сложности), «Результаты игр» и «Выход».

Данные о состоянии игрового процесса распределены по независимым глобальным массивам фиксированного размера 30*30 элементов, что исключает фрагментацию памяти:

1. Матрица board[MAX][MAX] — хранит цифровую карту поля (значение -1 зарезервировано под мину, числа от 0 до 8 указывают на количество мин вокруг ячейки).

2. Матрица revealed[MAX][MAX] — хранит логические флаги состояния видимости (открыта ли ячейка для игрока).

3. Матрица flagged[MAX][MAX] — содержит информацию о расставленных пользователем флагах-маркерах.

Обновление игрового процесса выполняется в асинхронном событийно-ориентированном цикле. Программа не расходует ресурсы процессора на пустые такты задержки Sleep(), а ожидает прерывания от системной очереди ввода с помощью блокирующих и неблокирующих функций WinAPI.

Для обеспечения расширенного взаимодействия с консолью Windows, обработки кликов мыши и интеграции низкоуровневых ассемблерных компонентов в проекте задействованы следующие функции Windows API:

— SetConsoleCursorPosition() — позиционирование курсора для обновления экрана без мерцания (аналог функции gotoxy).

— SetConsoleCP() и SetConsoleOutputCP() — установка кодировок ввода-вывода.

— GetConsoleMode() и SetConsoleMode() — чтение и модификация режимов консоли (используется для принудительного отключения режима быстрого редактирования ENABLE_QUICK_EDIT_MODE и активации флага обработки мыши ENABLE_MOUSE_INPUT).

— GetNumberOfConsoleInputEvents() — проверка наличия необработанных событий в системном буфере.

— ReadConsoleInput() — извлечение структур INPUT_RECORD и MOUSE_EVENT_RECORD для точного определения координат клика мыши и состояния её кнопок.

— FlushConsoleInputBuffer() — принудительная очистка буфера ввода для исключения «фантомных» кликов при переходах между игровыми экранами и меню.

— GetAsyncKeyState() — опрос аппаратного состояния клавиш клавиатуры (вызывается внутри ассемблерной вставки __asm для мгновенного перехвата нажатия клавиши P, отвечающей за паузу).

Выбранная многомодульная архитектура проекта (main.c, engine.c, ui.c, records.c) полностью себя оправдывает: логика обработки расстановки мин отделена от функций рендеринга символов интерфейса, а подсистема сохранения рекордов инкапсулирована и не зависит от текущего состояния графического буфера. Это обеспечивает высокую масштабируемость, простоту отладки и читаемость исходного кода.

Описание разработки программы

Разработка программного обеспечения выполнялась поэтапно на основе принципов модульного программирования. На первом этапе была определена общая высокоуровневая структура приложения: построение интерактивного главного меню, алгоритмы инициализации матриц игрового поля, организация основного игрового цикла с обработкой прерываний от мыши, а также подсистема долгосрочного хранения результатов. После этого были выделены ключевые глобальные переменные и сформирован сквозной заголовочный файл common.h.

Финальный состав многомодульного проекта распределен по следующим файлам:

— common.h — заголовочный файл, содержащий макроопределения констант, директивы препроцессора и объявления глобальных переменных с ключевым словом extern для сквозного доступа из разных модулей без дублирования памяти.

— main.c — файл реализации, содержащий точку входа в программу, первичную инициализацию системных дескрипторов консоли и главный цикл обработки выбора пользователя в меню.

— engine.h / engine.c — заголовок и файл реализации математической логики «Сапёра»: генерация мин, рекурсивный обход ячеек и управление сессией.

— ui.h / ui.c — заголовок и файл реализации графического рендеринга символьного интерфейса, а также низкоуровневой работы с буфером ввода Windows API.

— records.h / records.c — заголовок и файл реализации подсистемы файлового ввода-вывода для ведения таблицы рекордов.

Файл common.h инкапсулирует в себе базовые параметры конфигурации. В нем зафиксирована константа MAX = 30, определяющая

предельный размер двумерных массивов. Также здесь объявлены внешние переменные состояния: rows и cols (текущие габариты поля), mines (число скрытых угроз), opened_cells (счетчик успешно очищенных ячеек), game_over (триггер финала сессии) и game_active (флаг для фиксации режима паузы). Использование *includeguards* (#ifndef COMMON_H) предотвращает повторное включение заголовка при компиляции.

Файл main.c управляет общим жизненным циклом приложения. Функция main() настраивает локализацию консоли (кодировка 1251) и запрашивает дескриптор ввода. В бесконечном цикле while(1) происходит отрисовка главного меню. Модуль анализирует ввод пользователя через функцию scanf(). В зависимости от выбранного пункта меню, main.c осуществляет диспетчеризацию управления: либо возобновляет приостановленную игру, либо запускает подменю выбора сложности с последующей очисткой очередей ввода, либо вызывает экран рекордов, либо корректно завершает процесс.

Модуль engine.c реализует расчетную логику игры и состоит из трех ключевых функций:

— init_game(intdifficulty) — масштабирует игровое поле под выбранный уровень сложности, обнуляет рабочие матрицы, инициализирует генератор случайных чисел srand() и выполняет итеративную расстановку мин (значение -1 в массиве board). После этого функция математически высчитывает количество мин вокруг каждой безопасной ячейки и записывает эти значения в смежные ячейки.

— reveal(int r, int c) — «сердце» алгоритмической части программы. Это рекурсивная функция, которая открывает выбранную ячейку. Если ячейка пустая (вокруг нее 0 мин), функция автоматически запускает волновой обход во всех восьми направлениях, открывая смежные безопасные зоны до тех пор, пока не упрется в цифровые границы.

— play_game() — управляет итерациями внутриигрового цикла. Функция переводит консоль в режим чтения мыши, опрашивает буфер на наличие событий MOUSE_EVENT и, в зависимости от координат клика и нажатой кнопки (левая/правая), вызывает функции открытия ячеек или установки флагов в матрице flagged. Также здесь обрабатывается условие победы или подрыва на мине.

Модуль ui.c отвечает за визуальное представление данных и низкоуровневый опрос аппаратуры:

— draw_board() — осуществляет поцикловуюотрисовку координатной сетки и текущего состояния ячеек на основе данных из массивов board, revealed и flagged. Для исключения мерцания экрана вывод начинается строго из нулевой координаты консоли.

— gotoxy(int x, int y) — перемещает курсор в заданную позицию консоли с помощью WinAPI структуры COORD и функции SetConsoleCursorPosition().

— clear_screen() — выполняет системный вызов для очистки терминала.

— is_mouse_button_pressed_asm(intbutton) — низкоуровневый компонент, использующий вставку на языке ассемблера (__asm). Она напрямую загружает переданный код кнопки в регистр EAX, помещает его в стек и вызывает системную функцию GetAsyncKeyState, возвращая аппаратный статус нажатия. Данное решение гарантирует мгновенный перехват клавиши паузы P в обход стандартных задержек консольного буфера.

Модуль records.c изолирует в себе работу с файловой системой. Функция save_highscore() открывает текстовый файл highscores.txt в режиме добавления данных ("a") и выполняет форматированную запись имени игрока и его финального счета. Функция show_highscores() осуществляет потоковое

чтение файла строки за строкой с помощью fgets() и последовательно выводит таблицу лидеров на экран.

По порядку проектирования и реализации проекта можно сделать однозначный вывод об успешном применении метода нисходящего проектирования. Разработка продвигалась от абстрактной глобальной логики (структура меню и переключение экранов) к детальной реализации низкоуровневых подзадач (захват мыши, рекурсивный обход матриц, ассемблерные прерывания). Разделение кода на логические модули позволило изолировать интерфейс от математических расчетов, существенно упростило процесс отладки программы и повысило общую отказоустойчивость приложения.

Отладка и тестирование

Отладка приложения выполнялась в среде MicrosoftVisualStudio с использованием встроенного дебаггера: пошагового выполнения инструкций (F10/F11), расстановки точек останова (*breakpoints*) и мониторинга состояния матриц board, revealed и flagged через окно *Watch*.

Основное внимание уделялось корректности рекурсивного обхода пустых зон функции reveal(), точности трансляции координат клика мыши в индексы массивов поля и стабильности работы ассемблерной вставки при вызове паузы.

Сценарии тестирования

Для верификации программы был успешно выполнен следующий комплекс тест-кейсов:

— Тест-кейс №1: Запуск и инициализация. Корректное отображение меню на русском языке без искажения символов (кодировка CP1251).

— Тест-кейс №2: Навигация по меню. Переключение разделов ввода и устойчивость к некорректным данным (буквы вместо цифр не вызывают зависания).

— Тест-кейс №3: Выбор сложности. Динамическая перестройка сетки поля (9*9, 16*16 или 30*16) и генерация точного числа мин (10, 40 или 99).

— Тест-кейс №4: Левый клик мыши. Открытие безопасной ячейки с отображением цифры угроз. При значении 0 срабатывает каскадное рекурсивное открытие зоны.

— Тест-кейс №5: Правый клик мыши. Установка и снятие маркера F на закрытых ячейках. Левый клик по зафлагованной ячейке блокируется.

— Тест-кейс №6: Взаимодействие с миной. Фиксация проигрыша при клике на мину: игра останавливается, поле полностью раскрывается символами *.

— Тест-кейс №7: Успешное очищение поля. Фиксация победы, блокировка игрового цикла и вывод запроса имени игрока для записи рекорда.

— Тест-кейс №8: Режим паузы. Нажатие клавиши P мгновенно возвращает пользователя в меню за счет низкоуровневой ассемблерной вставки, активируя скрытый пункт «Возобновить игру».

— Тест-кейс №9: Возобновление сессии. Выбор пункта «0» восстанавливает игру с точным сохранением всех массивов, флагов и открытых клеток.

— Тест-кейс №10: Таблица рекордов. Чтение и форматированный вывод данных из highscores.txt. При отсутствии файла — вывод корректного предупреждения.

Выявленные и устраненные ошибки

В процессе разработки были обнаружены и локализованы следующие дефекты:

1. Эффект «фантомных кликов»: Клик выбора сложности в меню переносился на первый кадр игры. Исправлено очисткой буфера через fflush(stdin) и FlushConsoleInputBuffer() перед запуском цикла.

2. Смещение координат мыши: Клик по полю обрабатывался неверно из-за текстовых отступов номеров строк. Проведена калибровка формулы: col = (X - 4) / 2; row = Y - 1;.

3. Зацикливание при вводе букв: Ввод некорректных символов в меню вел к бесконечному циклу scanf. Решено валидацией ввода и очисткой потока через while (getchar() != '\n');.

Результат тестирования: Экспериментально подтверждена полная работоспособность, стабильность и отказоустойчивость приложения. Взаимодействие модулей и обработка игровых событий выполняются корректно.

Описание программы

Исполняемый файл kursach2.exe является основным компонентом разработанного программного комплекса. После запуска приложения пользователь попадает на экран интерактивного главного меню. Из главного меню доступны следующие сценарии: запуск новой сессии, возобновление приостановленной игры (режим паузы), просмотр сохраненной таблицы рекордов и штатное завершение работы программы.

В Таблица 1приведено подробное описание элементов управления, задействованных в игровом процессе.

Таблица 1 – Описание элементов управления

Действие	Описание
Левый клик мыши (ЛКМ)	Выбор и открытие закрытой ячейки игрового поля. При обнаружении мины сессия завершается поражением; при обнаружении безопасной зоны ячейка отображает количество смежных угроз.
Правый клик мыши (ПКМ)	Установка или снятие маркера опасности (флага) на закрытой ячейке поля. Защищает помеченную ячейку от случайного открытия кликом ЛКМ.
Клавиша P	Мгновенный перевод текущей игровой сессии в режим паузы с сохранением состояния всех матриц данных и автоматическим возвратом в главное меню.
Клавиша Enter	Подтверждение ввода числовых значений (выбор пунктов меню, ввод уровня сложности) и фиксация имени игрока при сохранении рекорда.

Схема взаимодействия программы:

— main.c — запускает главный цикл меню и обрабатывает выбор пользователя;

— ui.c — отвечает за отрисовку игрового поля и считывает координаты кликов мыши;

— engine.c — инициализирует расстановку мин и обрабатывает игровую логику обхода ячеек;

— records.c — отвечает за считывание и сохранение результатов в файл таблиц лидеров;

— common.h — хранит глобальные переменные, размеры матриц и общие константы проекта.

На Рисунке 1представлена схема взаимодействия программы.

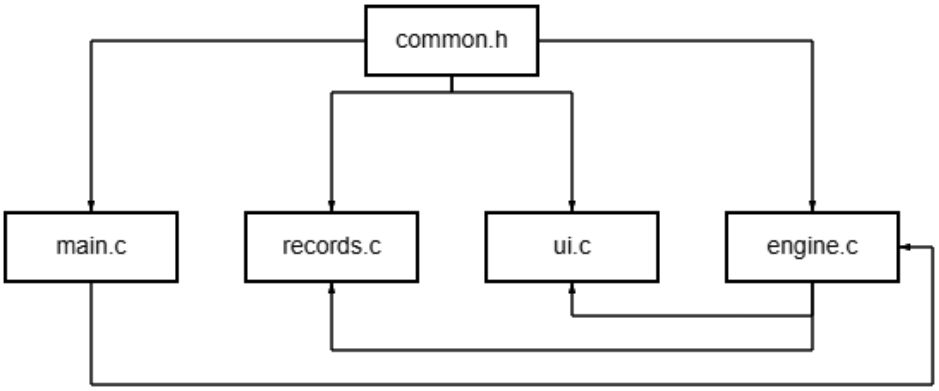


Рисунок 1- Схема взаимодействия программы

На Рисунке 2представлена схема данных.

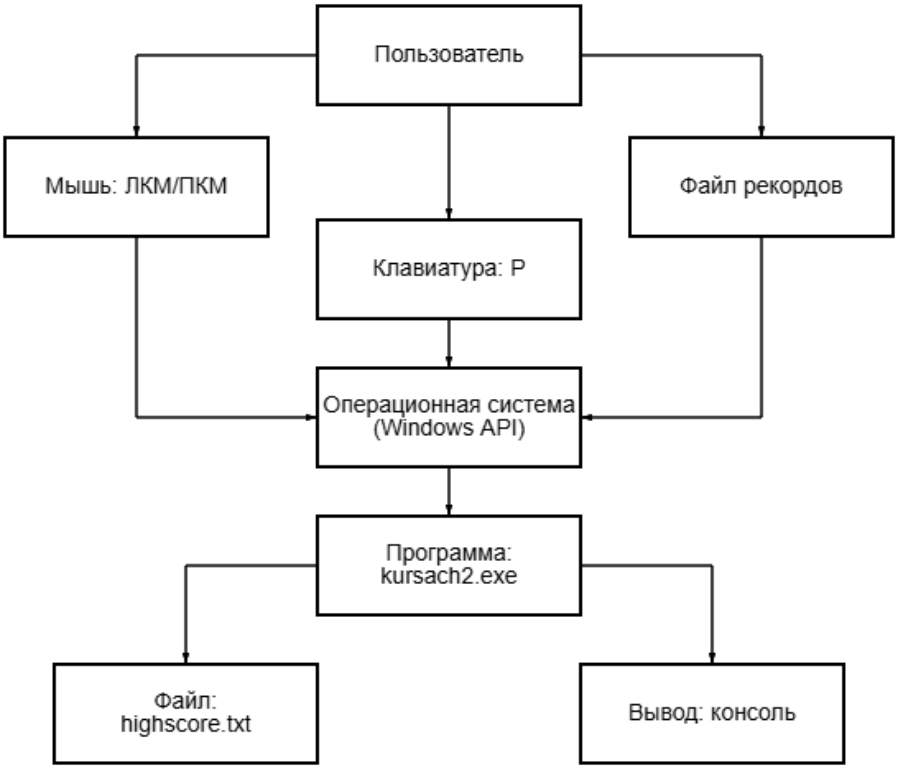


Рисунок 2 - Схема данных

Блок-схема Main представлена на Рисунке3, Рисунке4, Рисунке 5.23

Рисунок 3- Функция main

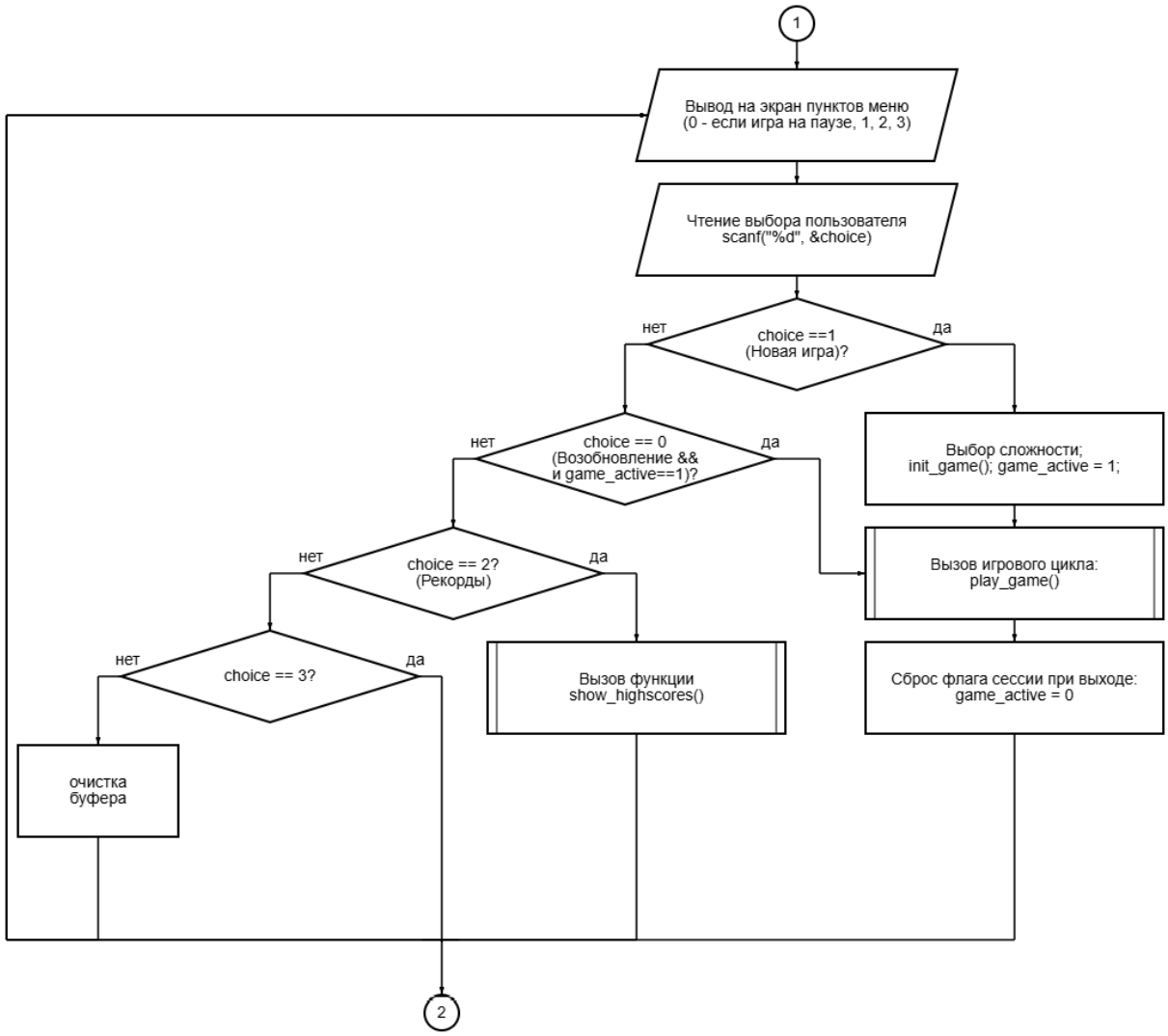


Рисунок 4 - Функция main

Рисунок 5 - Функция main

На Рисунке 6представлена блок-схема ассемблерной вставки из graphics.c.

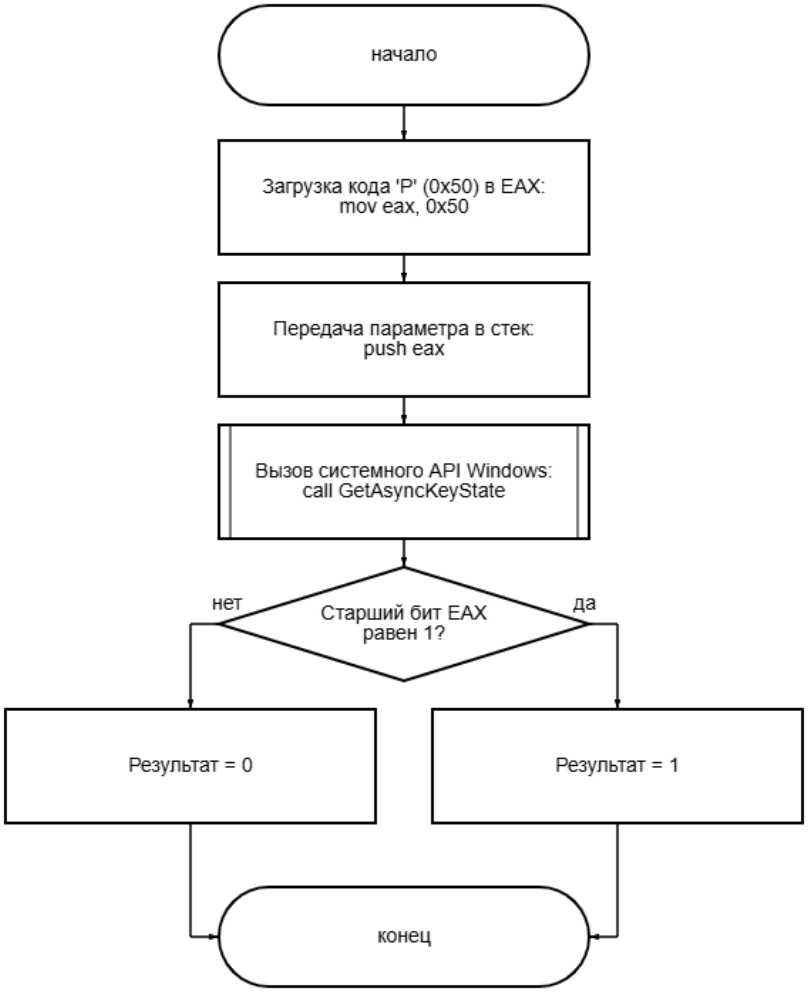


Рисунок 6 - Блок-схема ассемблерской вставки

Руководство пользователя

Программный комплекс предназначен для запуска консольной интерактивной игры «Сапёр». Для начала работы необходимо запустить исполняемый файл kursach2.exe.

После запуска программы открывается интерактивное (Рисунок 7) главное меню. Навигация по пунктам меню осуществляется путем ввода соответствующего цифрового символа с клавиатуры и подтверждения выбора нажатием клавиши Enter. Если в процессе выполнения игровой сессии была активирована пауза, в меню автоматически появится скрытый пункт «0. Возобновить игру».

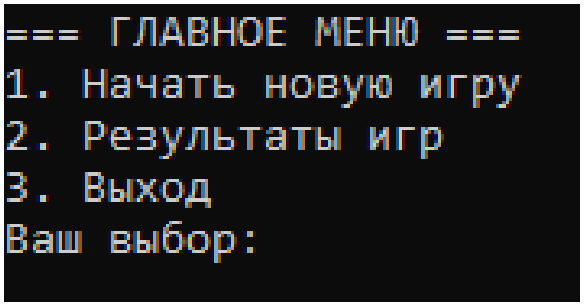


Рисунок 7 - главное меню

В главном меню для инициализации новой сессии необходимо выбрать пункт «1. Новая игра». После этого пользователю будет предложено настроить конфигурацию игрового поля, выбрав один из трех встроенных уровней сложности (Рисунок 8):

Легкий — поле 9*9, содержащее 10 скрытых мин.

Средний — поле 16*16, содержащее 40 скрытых мин.

Тяжелый — поле 30*16, содержащее 99 скрытых мин.

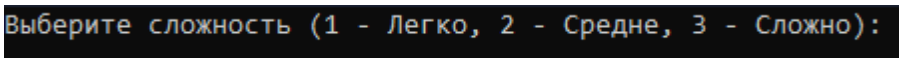


Рисунок 8 - выбор сложности

Основной игровой процесс визуализируется на графическом поле консоли. Управление полностью адаптировано под использование мыши.

При нажатии левой кнопки мыши (ЛКМ) на закрытую ячейку происходит ее открытие:

— Если в ячейке находится мина, игра мгновенно завершается поражением.

— Если мины нет, в ячейке отображается цифра, указывающая на количество угроз в смежных ячейках. Если вокруг ячейки мин нет, автоматически срабатывает рекурсивная функция открытия безопасной зоны.

При нажатии правой кнопки мыши (ПКМ) на закрытую ячейку пользователь устанавливает маркер опасности — флаг (обозначается символом F). Повторный клик ПКМ снимает маркер. Ячейка с флагом защищена от случайного открытия кликом ЛКМ.

Для оперативного перевода игры в режим паузы используется клавиша P на клавиатуре. При её нажатии состояние игрового поля замораживается, а пользователь возвращается в главное меню (Рисунок 9).

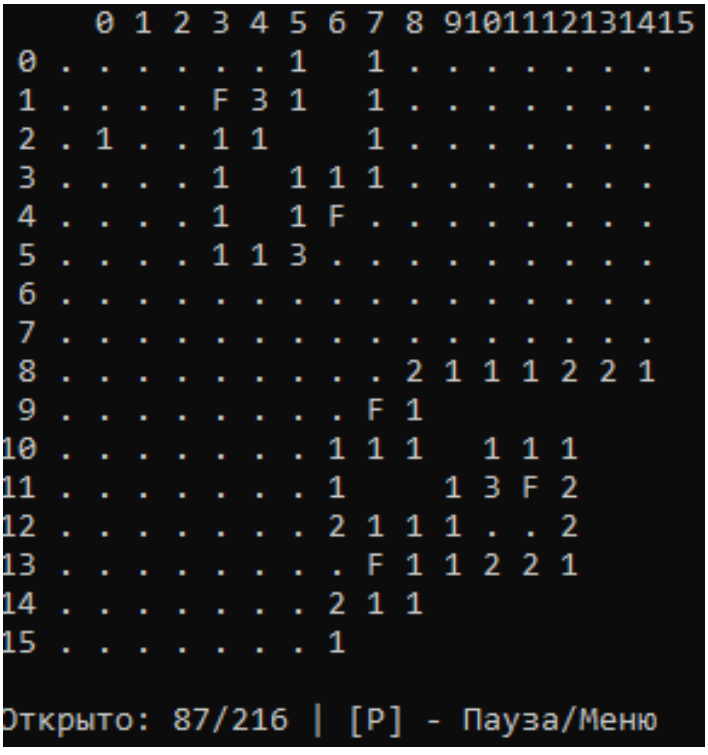


Рисунок 9 - игровое поле

Игра считается успешно пройденной (победа), если на поле открыты все безопасные ячейки, не содержащие мин. При наступлении победы или

поражения игра останавливается, открывая точное расположение всех мин на карте (мины обозначаются символом *) (Рисунок 10).

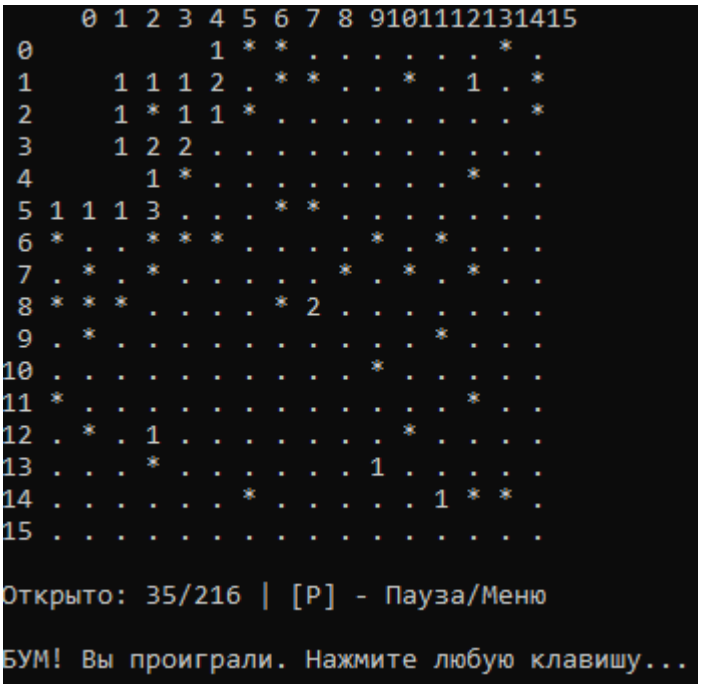


Рисунок 10 - процесс игровой сесии

В случае победы программа предлагает зафиксировать результат в глобальной таблице. Пользователю необходимо ввести имя игрока в консоли и нажать Enter. Данные автоматически форматируются и записываются в текстовый файл highscores.txt, который располагается в корневом каталоге программы (Рисунок 11).

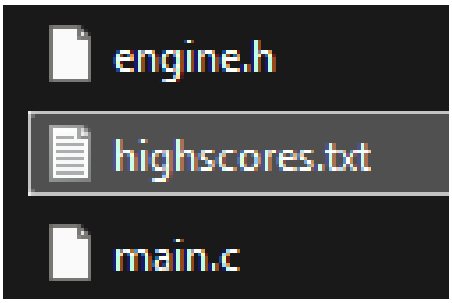


Рисунок 11 - файл highscores.txt

Для мониторинга лучших результатов необходимо в главном меню выбрать пункт «2. Рекорды». Программа считает данные из файла и выведет на экран структурированную таблицу лидеров (Рисунок 12),

отсортированную по времени прохождения или набранным очкам, с указанием имен игроков и выбранных уровней сложности.

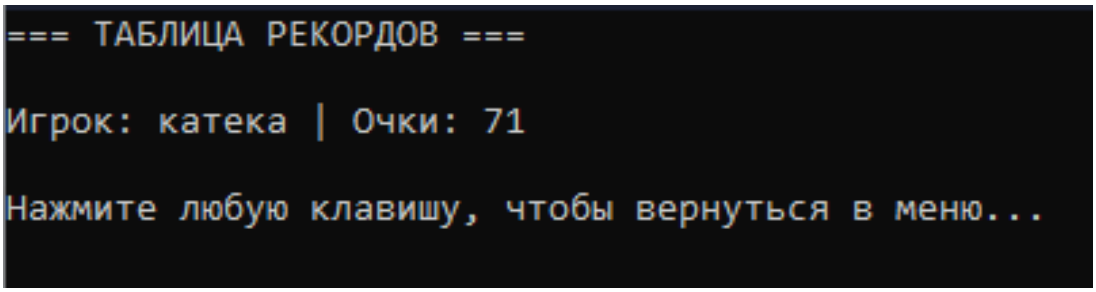


Рисунок 12 - окно просмотра рекордов

Заключение

В ходе выполнения курсовой работы была разработана интерактивная консольная игра «Сапёр» на языках программирования C и ASM. Программа успешно реализует классический расчетный алгоритм игрового процесса, адаптивный выбор уровней сложности, обработку манипуляций мыши через буфер консоли, режим паузы, сохранение результатов и структурированную таблицу лучших рекордов.

При разработке программного комплекса были закреплены практические навыки проектирования многомодульных приложений, работы со статическими двумерными массивами, структурами данных, текстовыми файлами, функциями низкоуровневого интерфейса Windows API и перехвата системных событий ввода. Также были успешно применены встроенные ассемблерные вставки (__asm) для оптимизации проверки состояния управляющих клавиш на уровне регистров процессора.

Разделение архитектуры приложения на отдельные функциональные модули (main.c, engine.c, ui.c, records.c) значительно упростило процессы поэтапного тестирования, отладки и верификации алгоритмов. Четкое разграничение зон ответственности каждого файла (логика генерации, интерфейс вывода, файловая система) сделало исходный код прозрачным и масштабируемым.

В качестве направлений для дальнейшего развития и модернизации разработанного программного продукта можно выделить следующие пункты:

— Добавление режима «Кастомной игры» с возможностью ручной настройки высоты, ширины поля и точного количества скрытых мин;

— Интеграция визуального таймера реального времени для фиксации скорости прохождения сессии;

— Реализация алгоритма «первого безопасного клика», гарантирующего, что при первом нажатии ЛКМ пользователь никогда не подорвется на мине;

— Внедрение звукового сопровождения игровых событий (клики, установка флага, победа, взрыв) средствами встроенного динамика;

— Обеспечение кроссплатформенной переносимости исходного кода для работы в операционных системах семейства Linux без жесткой привязки к библиотекам WinAPI.

Список используемых источников

1. Керниган Б., Ритчи Д. Язык программирования С. — 2-е изд. — М. : Вильямс, 2021. — 304 с.

2. MicrosoftLearn. Документация по WindowsConsole API [Электронный ресурс]. — URL: **https://learn.microsoft.com/ru-ru/windows/console/** (дата обращения: 01.05.2026).

3. MicrosoftLearn. Документация по языку С и компилятору MSVC [Электронный ресурс]. — URL: **https://learn.microsoft.com/ru-ru/cpp/c-language/** (дата обращения: 01.05.2026).

4. Павловская Т. А. С/С++. Программирование на языке высокого уровня. — СПб. : Питер, 2013. — 464 с.

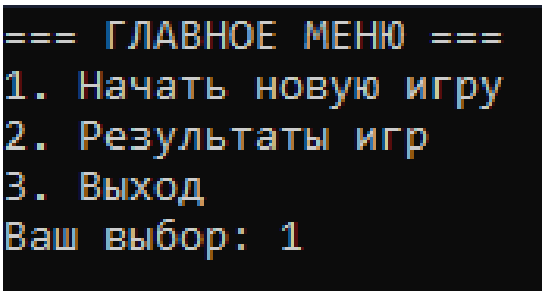
5. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 816 с.

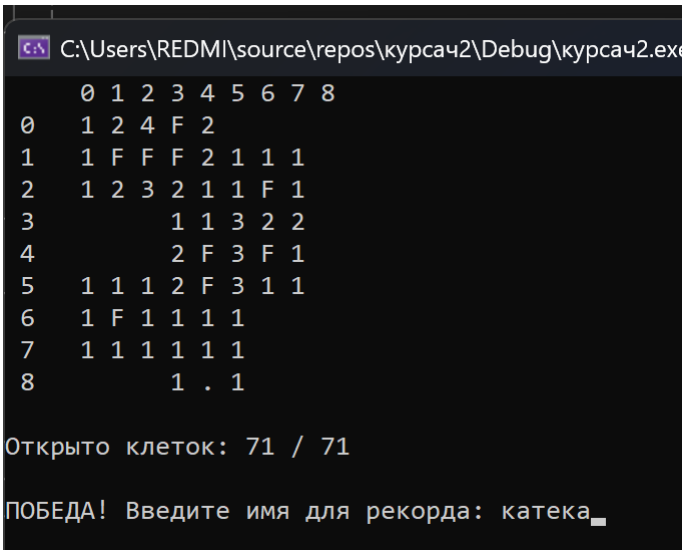
6. ГОСТ Р 2.105-2019. Единая система конструкторской документации. Общие требования к текстовым документам. — М. :Стандартинформ, 2019. — 42 с.

7. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. — М. : Издательство стандартов, 1990. — 26 с.

Приложение В. Результаты работы программы

Главное меню.



Выбор сложности.

Сохранение результатов.