Министерство высшего образования Российской федерации

Пензенский Государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование» на тему «Игра сапер»

Выполнил:

Студент группы 24ВВП1

Русанов А.Д.

Принял

к.т.н., доцент Токарев А.Н.

23.10.26

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 Русанову Александру Дмитриевичу Группа 24ВВП1

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

Исходные данные (технические требования) на проектирование

игра „Сапёр“
игрок играет с компьютером
Результаты игры сохраняются в файле
в программе присутствует ассемблерная вставка

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" Марта 2026

Дата защиты проекта "___" _____

Руководитель _____

Задание получил "18" Марта 2026 г.

Студент Русанов А.Д. _____

Содержание

Введение **6**
Постановка задачи **7**
Выбор решения **8**
Описание разработки программы **9**
Отладка и тестирование **10**
Описание программы **11**
Руководство пользователя **24**
Заключение **29**
Список используемых источников **30**
Приложение А **32**
Приложение В **41**

Введение

Компьютерные игры давно стали неотъемлемой частью цифровой культуры. Среди множества жанров особое место занимают логические игры, развивающие мышление, внимание и стратегическое планирование. Одной из таких игр является «Сапёр» — классическая игра-головоломка, известная многим пользователям персональных компьютеров с конца XX века.

История игры «Сапёр» уходит корнями в 1960–1970-е годы, когда появились первые текстовые и графические версии минных полей в настольных и консольных играх. Однако широкую известность она приобрела после включения в состав операционной системы Microsoft Windows, начиная с версии Windows 3.1, выпущенной в 1992 году. Игра поставлялась вместе с системой и служила не только для развлечения, но и как обучающий инструмент — она помогала пользователям привыкнуть к работе с мышью: научиться щелкать левой и правой кнопкой, а также взаимодействовать с элементами интерфейса.

Суть игры заключается в открытии ячеек поля, под которыми может находиться мина. Если игрок открыл ячейку с миной — он проигрывает. Однако если мины нет, на клетке появляется цифра, показывающая, сколько мин находится в восьми соседних клетках. Используя эти числовые подсказки, игрок логически определяет расположение всех мин и отмечает их флажками. Победа достигается, когда все мины найдены, и все остальные клетки — открыты.

В данной курсовой работе реализуется консольная версия игры «Сапёр» на языке программирования C++. Проект направлен на воссоздание базового функционала оригинальной игры с добавлением некоторых возможностей, таких как сохранение истории игр, отображение результатов и модульная организация кода.

Постановка задачи

Необходимо разработать программу игры «Сапёр». Интерфейс должен быть понятен и прост. Программа должна иметь интуитивно понятный интерфейс для пользователя.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы.

Использование сложных типов данных – массивов, структур, файлов. Это необходимо для более простой и интуитивной обработки данных в коде программы.

Режим работы видеосистемы – текстовый/графический. Для начала необходимо определиться с типом интерфейса и с элементами управления, затем необходимо изучить способы их реализации.

Устройство ввода-вывода – клавиатура и мышь. Необходимо различать и идентифицировать действия, произведенные с их помощью, это облегчит использование программы.

Интерфейс должен быть построен на основе меню. Это необходимо для создания интуитивно понятного интерфейса для пользователя.

Выбор решения

При запуске программы появляется меню, состоящее из четырех пунктов:

1. Запускает новую игру.
2. Позволяет изменить цвет фона и текста консоли.
3. Отображает список всех сыгранных игр.
4. Завершает выполнение программы.

Описание разработки программы

Разработанная программа состоит из нескольких модулей, которые реализовывались в следующем порядке:

1. Main.cpp – точка входа в программу.
2. Game.cpp – логика игры «Сапёр».
3. Menu.cpp – реализация меню и дополнительных функций.
4. Header.h – заголовочный файл с объявлениями всех функций и структур.

По порядку реализации модулей можно понять, что использовался нисходящей подход реализации программы. Этот метод основан на том, что сначала проектируются основные компоненты программы, а затем уже дорабатываются ее мелкие детали. Это позволяет упростить ее тестирование, так как каждый модуль тестируется сразу же после его реализации, что помогает в создании более структурированных программ.

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных.

Тестирование проводилось во время разработки и также после завершения разработки. В ходе нее было выявлено огромное количество проблем, связанных с работой с файлами, работой с памятью, работой с полями структуры.

Описание программы

Приложение main.exe является основным модулем программы. При запуске программы пользователю необходимо выбрать пункт меню для дальнейшей игры. Выход из меню осуществляется после выбора пользователем нужного пункта. Описание состояний программы выполнено в таблице 1.

Таблица 1 – Функция showMainMenu.

Клавиши, вызывающее событие	Действие пользователя	Действие программы
1,Enter	Выбран пункт «Start New Game»	Запуск новой игр
2,Enter	Выбран пункт «Change Console Color»	) Позволяет изменить цвет фона и текста консоли
3,Enter	Выбран пункт «View Game History»	) Отображает список всех сыгранных игр
4,Enter	Выбран пункт «Exit»	Выход из приложения

Таблица 2 описывает работу функции chooseConsoleColor, которая предоставляет пользователю возможность изменить цвет консоли. При выборе этого пункта меню пользователю предлагается ввести двухсимвольное шестнадцатеричное значение, где первый символ обозначает цвет фона, а второй − цвет текста.

Таблица 2 – Функция chooseConsoleColor.

Клавиши, вызывающее событие	Действие пользователя	Действие программы
0…F_{16},Enter	Выбор цвета	Изменяется цвет консоли

Таблица 3 описывает поведение функции next_move, отвечающей за получение хода от пользователя во время игры. Игрок вводит координаты ячейки, которую хочет открыть, в формате "строка столбец" – оба значения в пределах от 1 до 9. Функция проверяет корректность ввода: если введены недопустимые значения или уже открытая ячейка, программа выводит соответствующее сообщение об ошибке и запрашивает ввод повторно.

Таблица 3 – Функция next_move.

Клавиши, вызывающее событие	Действие пользователя	Действие программы
0…9,Enter	Выбор открывающейся ячейки	Открывается нужная ячейка

Далее на рисунке 1 представлена схема взаимодействия программы.

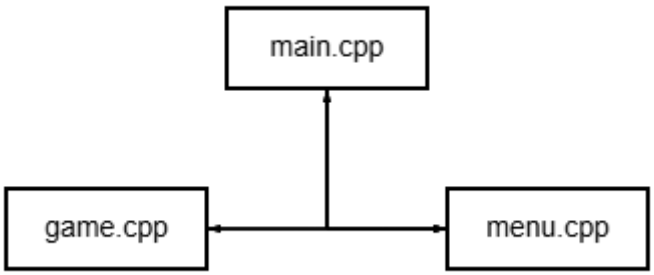


Рисунок 1 – Схема взаимодействия программы.

Ниже на рисунке 2 представлена схема данных.

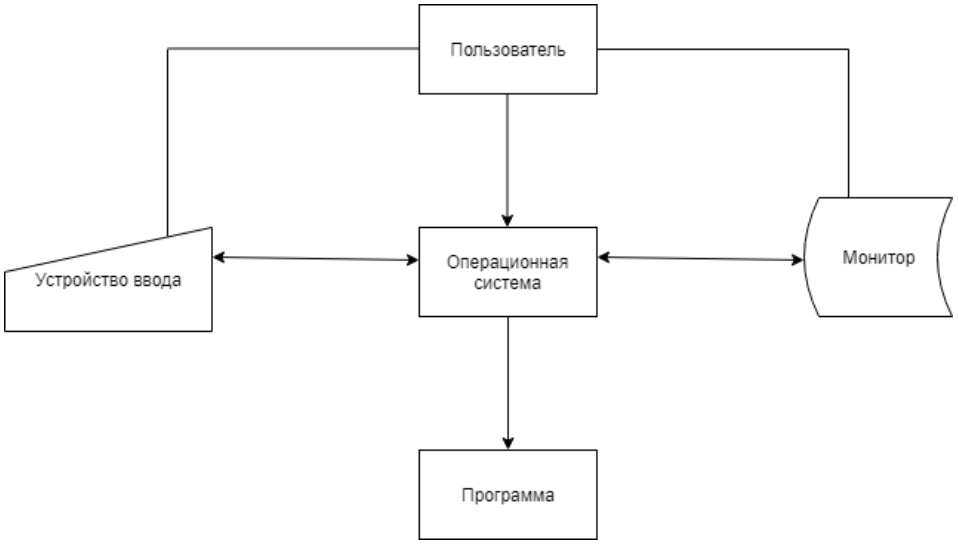


Рисунок 2 – Схема данных.

На рисунке 3 представлена схема программы. Функция Main.

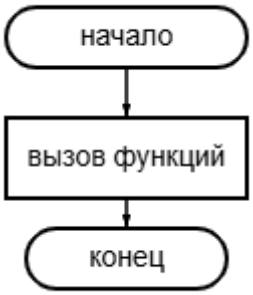


Рисунок 3 – Функция main.

На рисунке 4 – 5 представлена схема программы. Функция fill_board инициализирует игровое поле: очищает массивы и случайным образом расставляет 10 мин, обновляя значения соседних ячеек.

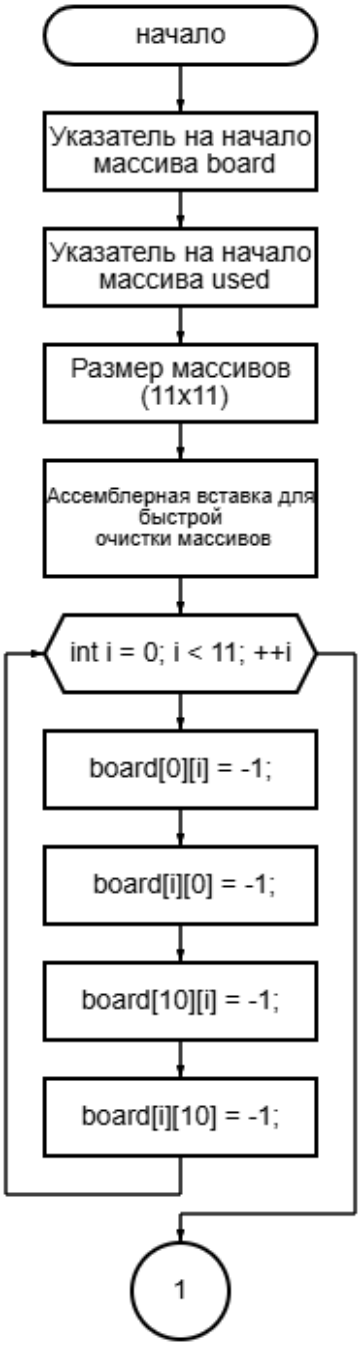


Рисунок 4 – Функция fill_board.

Рисунок 5 – Функция fill_board.

На рисунке 6 представлена схема программы. Функция next_move запрашивает у игрока координаты клетки для открытия и проверяет корректность ввода. При правильном вводе координат функция возвращает их для дальнейшей обработки хода в игре.

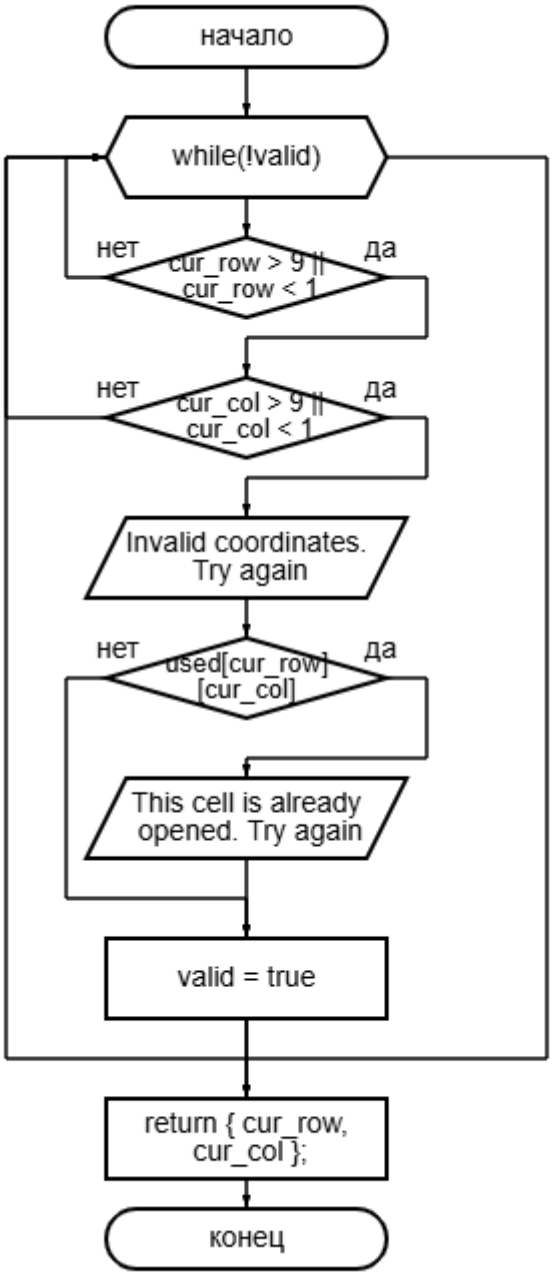


Рисунок 6 – Функция next_move.

На рисунке 7 – 8 представлена схема программы. Функция game.

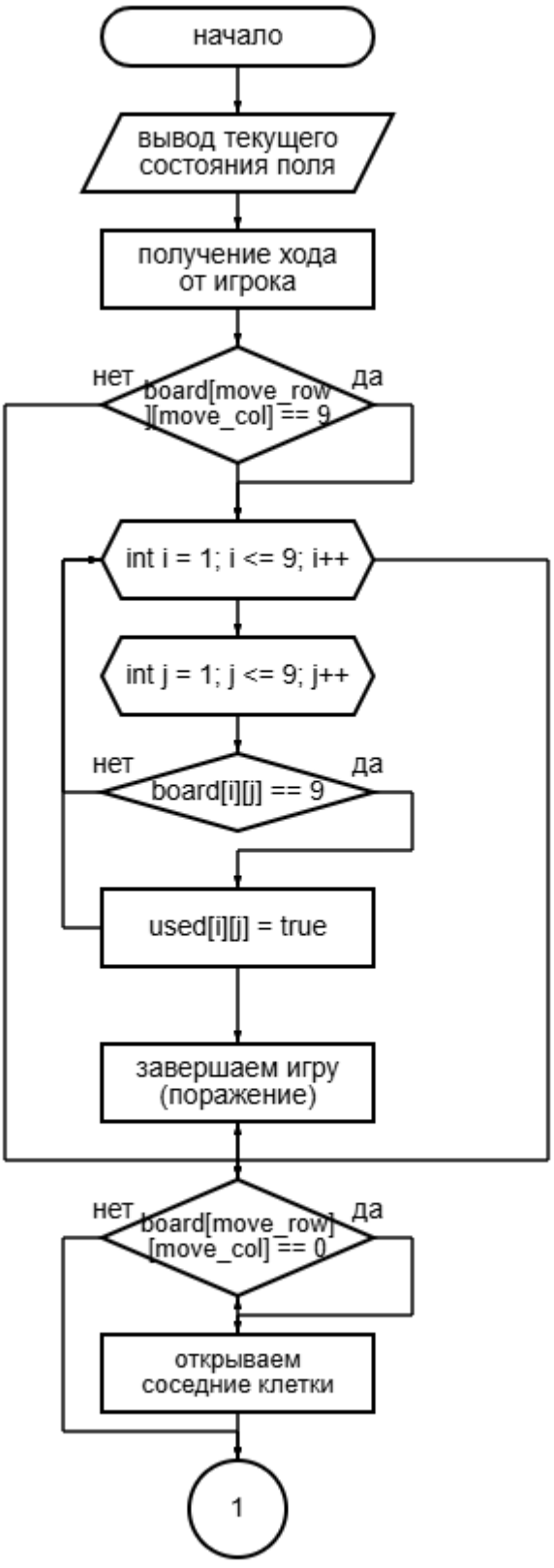


Рисунок 7 – Функция game.

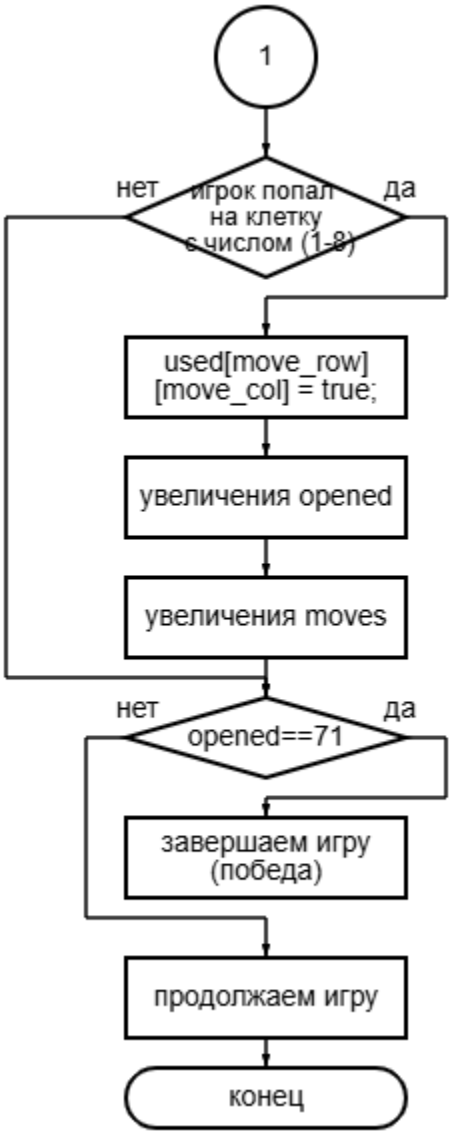


Рисунок 8 – Функция game.

На рисунке 9 представлена схема программы. Функция showMainMenu отображает главное меню игры и обрабатывает выбор пользователя. В зависимости от введённого номера меню запускается новая игра, настраивается цвет консоли, отображается история игр или происходит выход из программы.

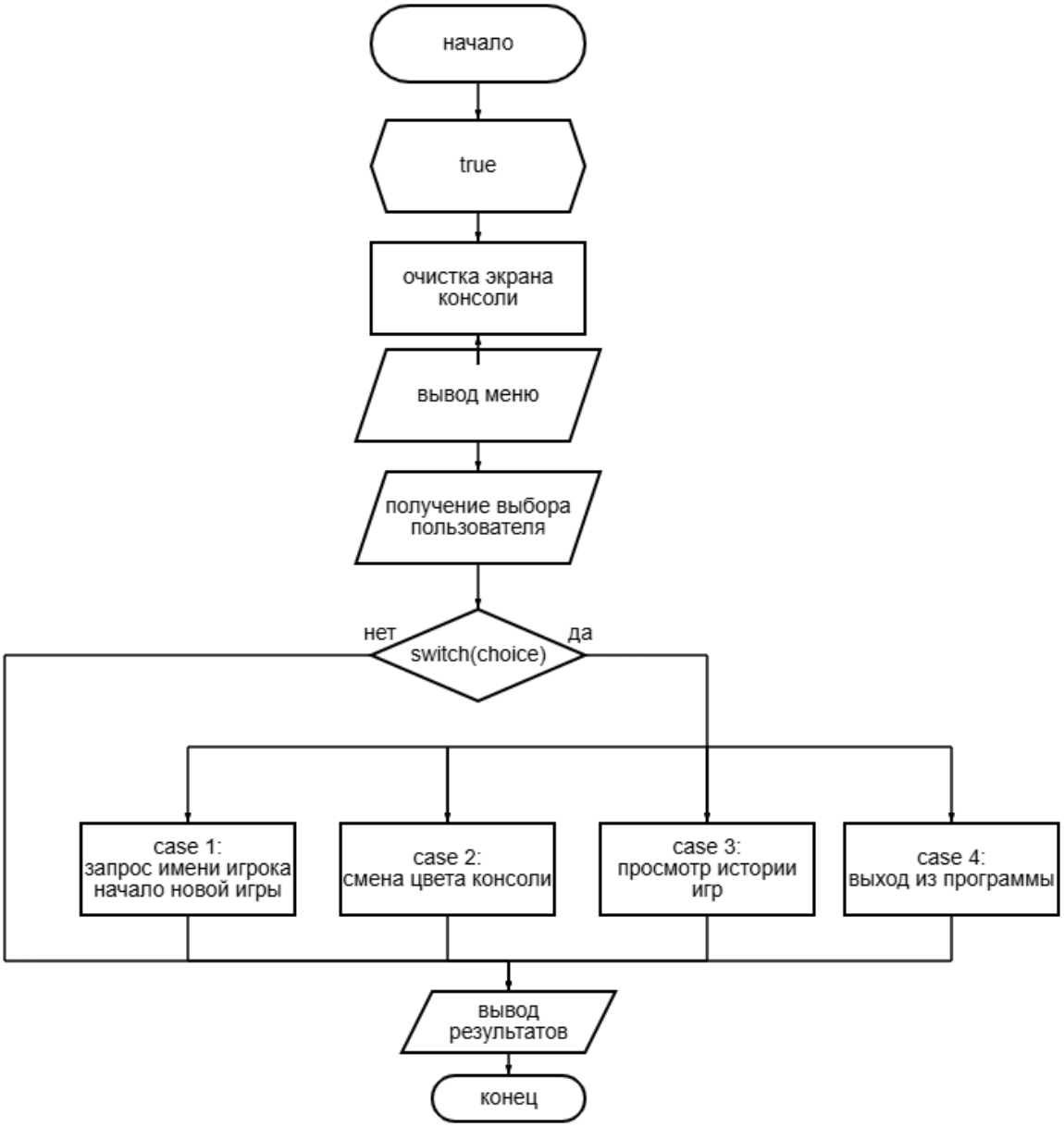


Рисунок 9 - Функция showMainMenu.

На рисунке 10 представлена схема программы. Функция loadGameHistory загружает историю ранее сыгранных игр из файла game_history.txt. Она считывает строки, извлекает из них данные о игроке, результате, количестве ходов и дате, и сохраняет эту информацию в вектор gameHistory.

Рисунок 10 - Функция loadGameHistory.

На рисунке 11 представлена схема программы. Функция showGameHistory выводит на экран список всех ранее сыгранных игр, загруженных в вектор gameHistory. Если история пуста, отображается соответствующее сообщение, в противном случае выводятся данные о каждом игроке, результате, количестве ходов и дате игры.

Рисунок 11 - Функция showGameHistory.

На рисунке 12 представлена схема программы. Функция chooseConsoleColor позволяет пользователю изменить цвет фона и текста консоли, вводя два шестнадцатеричных символа. При корректном вводе применяется соответствующая цветовая схема, в противном случае выводится сообщение об ошибке и повторный запрос ввода.

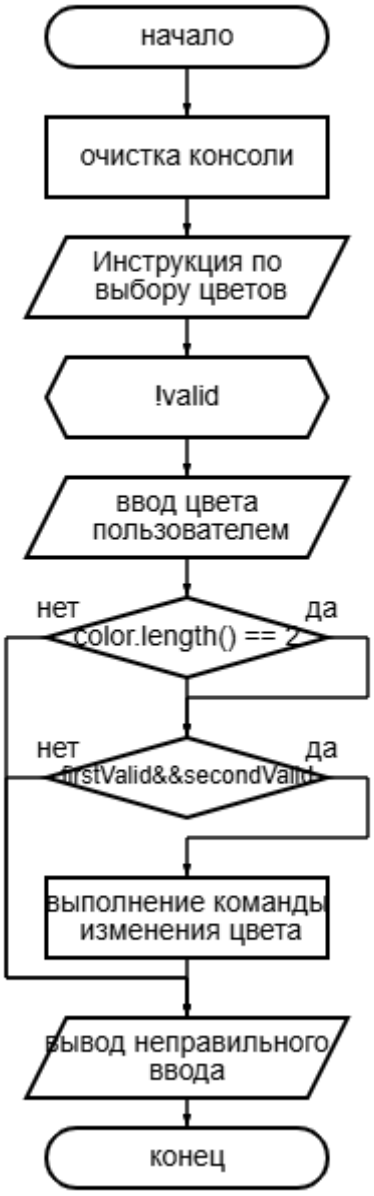


Рисунок 12 - Функция chooseConsoleColor.

Ассемблерные вставки в программе

В программе «Сапёр» используются ассемблерные вставки для очистки игрового поля и изменения значений счётчиков. Вставки реализованы в файле game.cpp с использованием конструкции __asm.

1. Очистка игрового поля

Первая ассемблерная вставка расположена в функции fill_board(). Она предназначена для обнуления массивов игрового поля и сброса счётчиков ходов и открытых клеток.

```
int* b = &board[0][0];
bool* u = &used[0][0];
int s = 121;

__asm {
    mov moves, 0
    mov opened, 0
    mov ecx, s
    mov edi, b
    mov esi, u
    xor eax, eax
clear_loop:
    mov [edi], eax
    mov [esi], al
    add edi, 4
    inc esi
    loop clear_loop
}
```

В данном фрагменте регистр ECX используется для подсчёта повторений цикла, EDI указывает на текущий элемент массива board, а ESI — на текущий элемент массива used. Регистр EAX обнуляется командой xor eax, eax. В цикле элементы обоих массивов последовательно заполняются нулевыми значениями. Цикл выполняется 121 раз.

2. Увеличение счётчика открытых клеток

Вторая ассемблерная вставка используется для увеличения значения переменной opened на единицу.

```
__asm {
    mov eax, opened
    add eax, 1
    mov opened, eax
}
```

Команда mov eax, opened загружает текущее значение счётчика в регистр EAX. Команда add eax, 1 увеличивает его на единицу, после чего mov opened, eax сохраняет результат в переменной opened.

3. Увеличение счётчика ходов

Третья ассемблерная вставка предназначена для увеличения значения переменной moves, которая хранит количество выполненных ходов.

```
__asm {
    mov eax, moves
    add eax, 1
    mov moves, eax
}
```

Сначала текущее значение moves помещается в регистр EAX. Затем оно увеличивается на единицу и записывается обратно в переменную. Таким образом, программа учитывает количество выполненных пользователем ходов.

Вывод

Ассемблерные вставки применяются в программе для выполнения операций с массивами игрового поля и счётчиками. Первая вставка очищает игровое поле и сбрасывает счётчики, вторая увеличивает количество открытых клеток, а третья — количество ходов. Остальная логика игры реализуется средствами языка C++

Руководство пользователя

Программа Minesweeper.exe представляет собой реализацию популярной логической игры *Сапёр* и предназначена для тренировки стратегического мышления, внимания и памяти. Игра выполнена в консольном интерфейсе, удобна для использования и не требует специальной подготовки.

После запуска приложения на экране появляется главное меню (рисунок 13), в котором можно выбрать один из пунктов:

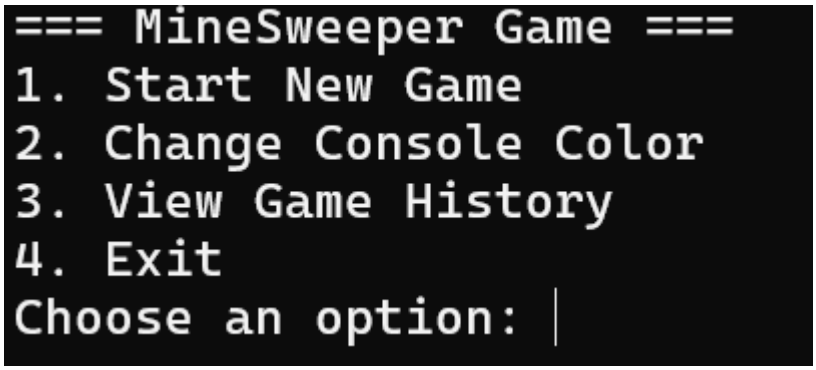


Рисунок 13 – Главное меню.

При выборе Change Console Color пользователю предлагается ввести двухсимвольное шестнадцатеричное значение, где первый символ — цвет фона, второй — цвет текста. Например, 1F — синий фон и белый текст. В случае неправильного ввода появится сообщение об ошибке и запрос будет повторён (рисунок 14).

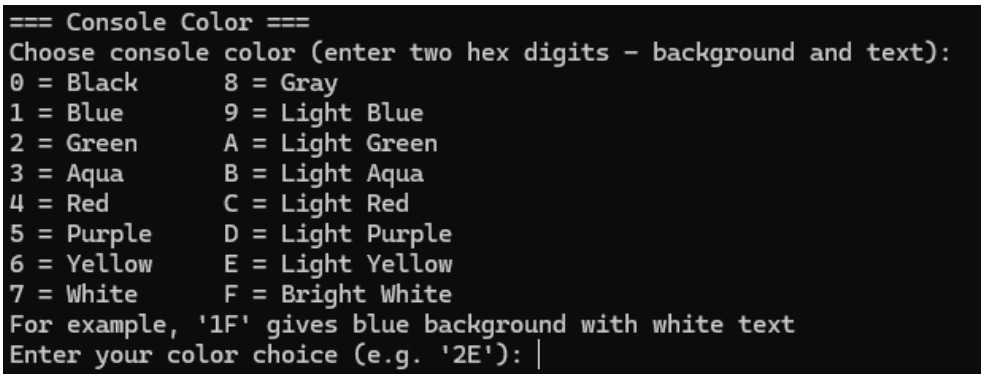


Рисунок 14 – Меню изменения цвета.

После выбора параметров, происходит отрисовка игры (рисунок 15).

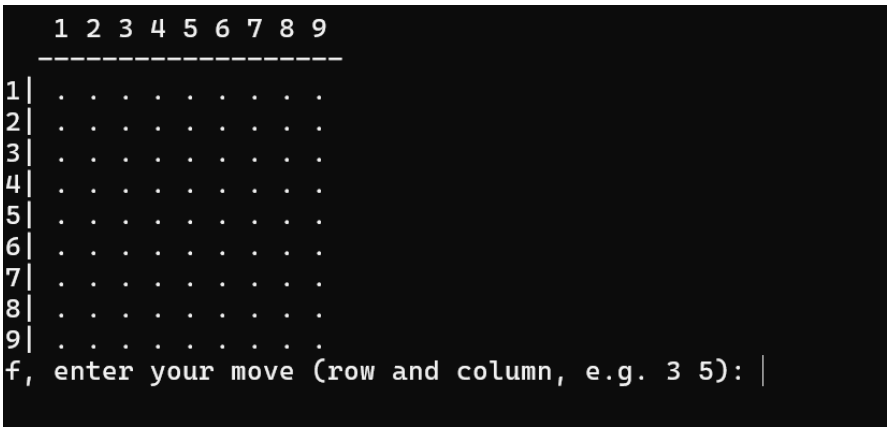


Рисунок 15 – Отрисовка игры.

Управление в игре осуществляется с помощью клавиатуры и состоит из ввода числовых команд в меню и координат клеток во время игры.

Вывод итогов игры представлен на рисунке 16.

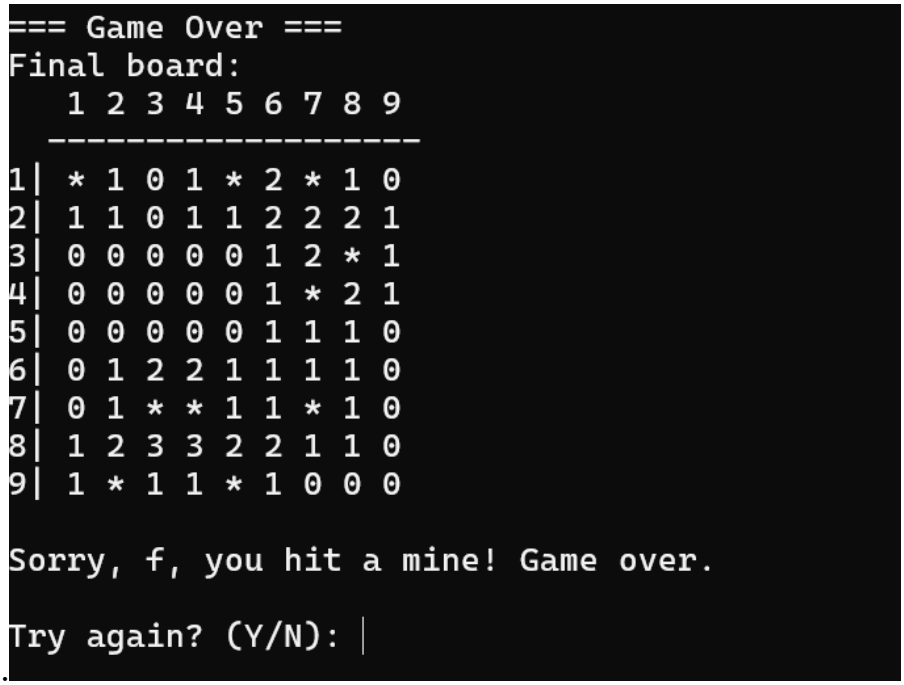


Рисунок 16 – Вывод результатов игры.

Пункт View Game History отображает список всех ранее сыгранных игр, включая имя игрока, результат (победа или поражение), количество ходов и дату игры. История сохраняется в файл game_history.txt (рисунок 17).

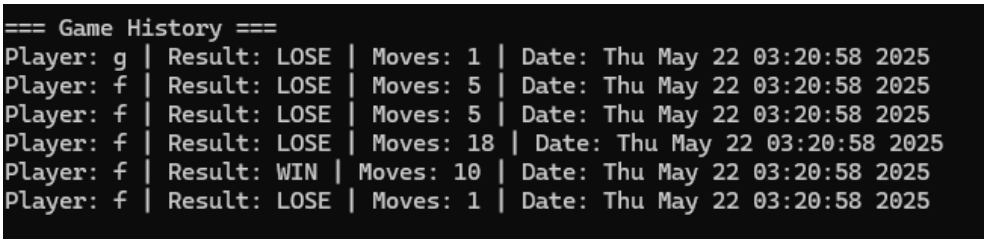


Рисунок 17 – Вывод результатов игры.

Заключение

При выполнении данной курсовой работы были получены навыкиразработки многомодульных программ. Была освоена работа с файлами, изучены функции работы с консолью и оперативной памятью. Также были получены основные навыки отладки и тестирования программ и программирования в среде Visual Studio 2022 на языках C/C++ и Assembler.

В рамках курсовой работы была написана программа с игрой «Сапёр». Она имеет все необходимые функции для получения игрового опыта в игре «Сапёр».

В дальнейшем программу можно улучшить, путем добавления графического интерфейса, что позволит улучшить навигацию по меню программы, путем добавления возможности использования мыши.

Список используемых источников

1. Аблязов, Р. З. Программирование на ассемблере для платформы x86-64: учебное пособие / Р. З. Аблязов. – М. : ДМК Пресс, 2016. – 392 с.
2. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт ; пер. с англ. – М. : Диалектика, 2017. – 272 с.
3. Гудрич, М. Т. Структуры данных и алгоритмы на C++ / М. Т. Гудрич, Р. Тамассия, М. Х. Голдвассер ; пер. с англ. – М. : Техносфера, 2018. – 880 с.
4. Дейтел, П. Как программировать на C++ / П. Дейтел, Х. Дейтел ; пер. с англ. – 10-е изд. – СПб. : Питер, 2017. – 1008 с.
5. Керниган, Б. В. Язык программирования C / Б. В. Керниган, Д. М. Ритчи ; пер. с англ. – 2-е изд., испр. и доп. – М. : Финансы и статистика, 1992. – 272 с.
6. Клири, Дж. C++: ускоренный курс / Дж. Клири ; пер. с англ. – СПб. : Питер, 2020. – 608 с.
7. Кормен, Т. Х. Алгоритмы: построение и анализ / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн ; пер. с англ. – 3-е изд. – М. : И.Д. Вильямс, 2013. – 1328 с.
8. Лафоре, Р. Объектно-ориентированное программирование в C++ / Р. Лафоре ; пер. с англ. – 4-е изд. – СПб. : Питер, 2016. – 928 с.
9. Липпман, С. Б. Язык программирования C++. Базовый курс / С. Б. Липпман, Ж. Лажойе, Б. Э. Му ; пер. с англ. – 5-е изд. – М. : И.Д. Вильямс, 2014. – 1120 с.
10. Прата, С. Язык программирования C++. Лекции и упражнения / С. Прата ; пер. с англ. – 6-е изд. – М. : ООО "И.Д. Вильямс", 2019. – 1248 с.
11. Седжвик, Р. Алгоритмы на C++. Фундаментальные алгоритмы, структуры данных, сортировка, поиск / Р. Седжвик, К. Уэйн ; пер. с англ. – 4-е изд. – М. : И.Д. Вильямс, 2016. – 1056 с.
12. Страуструп, Б. Тур по C++ / Б. Страуструп ; пер. с англ. – 2-е изд. – М. : ДМК Пресс, 2019. – 304 с.

13. Страуструп, Б. Язык программирования C++ / Б. Страуструп; пер. с англ. – Специальное изд. – М. : Бином-Пресс, 2011. – 1136 с.

14. Троелсен, Э. Программирование на C++. Подробное руководство / Э. Троелсен, Ф. Фёрниш ; пер. с англ. – 5-е изд. – М. : Диалектика, 2021. – 1392 с.

15. Шилдт, Г. C++: руководство для начинающих / Г. Шилдт ; пер. с англ. – 3-е изд. – М. : И.Д. Вильямс, 2015. – 720 с.

Приложение А

Листинги программы

Файл header.h

```
#ifndef MINESWEEPER_H
#define MINESWEEPER_H

#define _CRT_SECURE_NO_WARNINGS
#include <sstream>
#include <time.h>
#include <stdlib.h>
#include <iostream>
#include <cstdlib>
#include <string>
#include <vector>
#include <fstream>
#include <ctime>
using namespace std;

// Структура для хранения результатов игры
struct GameResult {
    string playerName;
    bool win;
    time_t date;
    int moves;
};

extern vector<GameResult> gameHistory; // Вектор для хранения
истории игр
extern string playerName; // Имя игрока
extern int board[11][11];
extern bool used[11][11];
extern int row[];
extern int col[];
extern int moves;
extern int opened;

// Функции меню
void showMainMenu();
void loadGameHistory();
void showGameHistory();
void chooseConsoleColor();
void enterPlayerName();

// Функции игры
void restart();
void game();
void fill_board();
void print_board();
pair<int, int> next_move();
```

```
void recursion(int move_row, int move_col);
void go_on(bool win);
void saveGameResult(bool win);
#endif
```

Файл main.cpp

```
#include "header.h"

int main() {
    loadGameHistory();
    showMainMenu();
    return 0;
}
```

Файл game.cpp

```
#include "header.h"

vector<GameResult> gameHistory;
string playerName;
int board[11][11];
bool used[11][11];
int row[] = { -1, -1,-1, 0,0, 1,1,1 };
int col[] = { -1, 0, 1, -1,1, -1, 0,1 };
int moves;
int opened;

void fill_board() {
    int* b = &board[0][0];
    bool* u = &used[0][0];
    int s = 121;

    __asm {
        mov moves, 0
        mov opened, 0
        mov ecx, s
        mov edi, b
        mov esi, u
        xor eax, eax
        clear_loop :
        mov[edi], eax
            mov[esi], al
            add edi, 4
            inc esi
            loop clear_loop
    }

    for (int i = 0; i < 11; ++i) {
        board[0][i] = -1;
```

```
        board[i][0] = -1;
        board[10][i] = -1;
        board[i][10] = -1;
    }

    srand(time(0));
    int mines = 0;
    while (mines < 10) {
        int x = rand() % 9 + 1;
        int y = rand() % 9 + 1;
        if (board[x][y] != 9) {
            board[x][y] = 9;
            for (int i = 0; i < 8; ++i) {
                if (board[x + row[i]][y + col[i]] != -1 && board[x +
row[i]][y + col[i]] != 9) {
                    board[x + row[i]][y + col[i]]++;
                }
            }
            mines++;
        }
    }
}

void print_board() {
    cout << "   ";
    for (int i = 1; i <= 9; i++) cout << i << " ";
    cout << "\n  --------------------\n";

    for (int i = 1; i <= 9; i++) {
        cout << i << "| ";
        for (int j = 1; j <= 9; j++) {
            if (used[i][j]) {
                if (board[i][j] == 9) cout << "* ";
                else cout << board[i][j] << " ";
            }
            else {
                cout << ". ";
            }
        }
        cout << "\n";
    }
}

pair<int, int> next_move() {
    bool valid = false;
    int cur_row, cur_col;
    while (!valid) {
        cout << playerName << ", enter your move (row and column,
e.g. 3 5): ";

        if (!(cin >> cur_row >> cur_col)) {
```

```
            cout << "Invalid input. Please enter two numbers.\n";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }

        if (cur_row > 9 || cur_row < 1 || cur_col > 9 || cur_col <
1) {
            cout << "Invalid coordinates. Try again.\n";
        }
        else if (used[cur_row][cur_col]) {
            cout << "This cell is already opened. Try again.\n";
        }
        else {
            valid = true;
        }
    }
    return { cur_row, cur_col };
}

void recursion(int move_row, int move_col) {
    used[move_row][move_col] = true;
    opened++;
    for (int i = 0; i < 8; ++i) {
        if (!used[move_row + row[i]][move_col + col[i]]) {
            if (board[move_row + row[i]][move_col + col[i]] > 0 &&
board[move_row + row[i]][move_col + col[i]] < 9) {
                used[move_row + row[i]][move_col + col[i]] = true;
                opened++;
            }
            else if (board[move_row + row[i]][move_col + col[i]] ==
0) {
                recursion(move_row + row[i], move_col + col[i]);
            }
        }
    }
}

void go_on(bool win) {
    system("cls");
    cout << "=== Game Over ===\n";
    cout << "Final board:\n";

    cout << "   ";
    for (int i = 1; i <= 9; i++) cout << i << " ";
    cout << "\n  -------------------\n";

    for (int i = 1; i <= 9; i++) {
        cout << i << "| ";
        for (int j = 1; j <= 9; j++) {
            if (board[i][j] == 9) cout << "* ";
```

```
            else cout << board[i][j] << " ";
        }
        cout << "\n";
    }

    if (win) {
        cout << "\nCongratulations, " << playerName << "! You won in
" << moves << " moves!\n";
        saveGameResult(true);
    }
    else {
        cout << "\nSorry, " << playerName << ", you hit a mine! Game
over.\n";
        saveGameResult(false);
    }

    cout << "\nTry again? (Y/N): ";
    char re;
    cin >> re;
    if (toupper(re) == 'Y') {
        restart();
    }
    else {
        cout << "\nReturning to main menu...\n";
        system("pause");
        showMainMenu();
    }
}

void game() {
    system("cls");
    print_board();
    pair<int, int> move = next_move();
    int move_row = move.first;
    int move_col = move.second;

    if (board[move_row][move_col] == 9) {
        for (int i = 1; i <= 9; i++) {
            for (int j = 1; j <= 9; j++) {
                if (board[i][j] == 9) used[i][j] = true;
            }
        }
        go_on(false);
    }
    else if (board[move_row][move_col] == 0) {
        recursion(move_row, move_col);
    }
    else {
        used[move_row][move_col] = true;
        __asm {
            mov eax, opened
```

```
            add eax, 1
            mov opened, eax
        }
    }

    __asm {
        mov eax, moves
        add eax, 1
        mov moves, eax
    }

    if (opened == 71) {
        go_on(true);
    }
    else {
        game();
    }
}

void restart() {
    fill_board();
    game();
}

void saveGameResult(bool win) {
    GameResult result;
    result.playerName = playerName;
    result.win = win;
    result.date = time(nullptr);
    result.moves = moves;
    gameHistory.push_back(result);

    ofstream outFile("game_history.txt", ios::app);
    if (outFile.is_open()) {
        outFile << "Player: " << playerName << " | Result: " << (win
? "WIN" : "LOSE")
            << " | Moves: " << moves << " | Date: " <<
ctime(&result.date);
        outFile.close();
    }
}
```

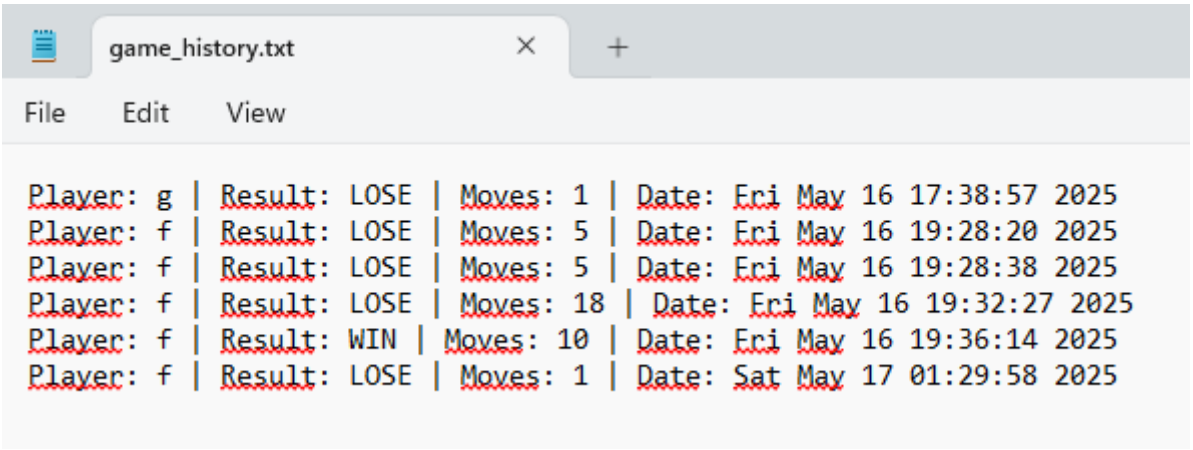
Файл menu.cpp

```
#include "header.h"

void showMainMenu() {
    while (true) {
        system("cls");
        cout << "=== MineSweeper Game ===\n";
        cout << "1. Start New Game\n";
        cout << "2. Change Console Color\n";
        cout << "3. View Game History\n";
```

```
        cout << "4. Exit\n";
        cout << "Choose an option: ";

        int choice;
        cin >> choice;

        switch (choice) {
        case 1:
            enterPlayerName();
            restart();
            break;
        case 2:
            chooseConsoleColor();
            break;
        case 3:
            showGameHistory();
            break;
        case 4:
            exit(0);
        default:
            cout << "Invalid choice. Please try again.\n";
            system("pause");
            break;
        }
    }
}

void loadGameHistory() {
    ifstream inFile("game_history.txt");
    if (!inFile.is_open()) return;

    string line;
    while (getline(inFile, line)) {
        GameResult result;
        size_t pos1 = line.find("Player: ");
        size_t pos2 = line.find(" | Result: ");
        size_t pos3 = line.find(" | Moves: ");
        size_t pos4 = line.find(" | Date: ");

        if (pos1 == string::npos || pos2 == string::npos || pos3 ==
string::npos || pos4 == string::npos)
            continue;

        result.playerName = line.substr(pos1 + 8, pos2 - (pos1 +
8));
        string resultStr = line.substr(pos2 + 11, pos3 - (pos2 +
11));
        result.win = (resultStr == "WIN");

        string movesStr = line.substr(pos3 + 9, pos4 - (pos3 + 9));
        try {
```

```
            result.moves = stoi(movesStr);
        }
        catch (...) {
            result.moves = 0;
        }

        string dateStr = line.substr(pos4 + 9);
        result.date = time(nullptr);
        gameHistory.push_back(result);
    }

    inFile.close();
}

void showGameHistory() {
    system("cls");
    cout << "=== Game History ===\n";

    if (gameHistory.empty()) {
        cout << "No games played yet.\n";
    }
    else {
        for (const auto& result : gameHistory) {
            cout << "Player: " << result.playerName << " | Result: "
<< (result.win ? "WIN" : "LOSE")
                 << " | Moves: " << result.moves << " | Date: " <<
ctime(&result.date);
        }
    }

    cout << "\nPress Enter to return to menu...";
    cin.ignore();
    cin.get();
}

void chooseConsoleColor() {
    system("cls");
    cout << "=== Console Color ===\n";
    cout << "Choose console color (enter two hex digits - background
and text):\n";
    cout << "0 = Black      8 = Gray\n";
    cout << "1 = Blue       9 = Light Blue\n";
    cout << "2 = Green      A = Light Green\n";
    cout << "3 = Aqua       B = Light Aqua\n";
    cout << "4 = Red        C = Light Red\n";
    cout << "5 = Purple     D = Light Purple\n";
    cout << "6 = Yellow     E = Light Yellow\n";
    cout << "7 = White      F = Bright White\n";
    cout << "For example, '1F' gives blue background with white
text\n";
```

```
    string color;
    bool valid = false;

    while (!valid) {
        cout << "Enter your color choice (e.g. '2E'): ";
        cin >> color;

        if (color.length() == 2) {
            bool firstValid = (color[0] >= '0' && color[0] <= '9')
||
                (toupper(color[0]) >= 'A' && toupper(color[0]) <=
'F');
            bool secondValid = (color[1] >= '0' && color[1] <= '9')
||
                (toupper(color[1]) >= 'A' && toupper(color[1]) <=
'F');

            if (firstValid && secondValid) {
                valid = true;
                string command = "color " + color;
                system(command.c_str());
            }
        }

        if (!valid) {
            cout << "Invalid color choice. Please try again.\n";
        }
    }
}

void enterPlayerName() {
    cout << "Enter your name: ";
    cin >> playerName;
}
```

Приложение В

Результат работы программы: