

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Пензенский государственный университет»
Факультет вычислительной техники
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. 24BBB1

Гурин А.Н.

Проверил: к.т.н. доцент

Генералова А.А.

*Проверено
04.06.2025
Г*

Пенза 2025

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

М.А. Митрохин

« 14 » февраля 2023 г.

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Гурину Алексею Николаевичу Группа 24ВВВ1

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

Исходные данные (технические требования) на проектирование

Игра "Сапер". Игра с компьютером.
Работа с файлами.
Три уровня сложности.

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 "14" 02.2025

Дата защиты проекта "04" июня 2025

Руководитель Г. Герасимов А.А.

Задание получил "14" 02 2025г.

Студент Гурин (Гурин Алексей Николаевич)

Содержание

Введение.....	7
1 Постановка задачи.....	8
2 Выбор Решения.....	9
3 Описание разработки программы.....	10
4 Отладка и тестирование.....	11
5 Описание программы.....	12
5.1 Схема функции main.....	12
5.2 Описание модулей.....	12
6 Руководство пользователя.....	15
6.1 Заставка	15
6.2 Правила пользования.....	15
6.3 Меню уровней	16
6.5 Открытие клетки	17
6.6 Аварийный выход	17
6.7 Сохранение результата при аварийном выводе.....	18
6.8 Поражение.....	18
6.9 Сохранение результата при поражении.....	19
6.10 Победа	19
6.11 Сохранение результата при победе.....	20
Заключение	21
Список используемых материалов	22
Приложение А	23
Листинг программы	23
Файл sapper.c.....	23
Файл splash.c	24
Файл splash.h.....	25
Файл board.c	25
Файл board.h.....	29
Файл game.c	30
Файл game.h	33

Введение

В современном мире компьютерные игры занимают значительное место в сфере развлечений, образования и даже профессиональной деятельности. Одной из классических и популярных игр является «Сапёр», которая не только развивает логическое мышление, но и представляет интерес с точки зрения алгоритмизации и программирования. Изучение принципов работы этой игры позволяет углубить знания в области разработки программного обеспечения, обработки пользовательского ввода и генерации игровых полей.

Актуальность работы заключается в том, что реализация игры «Сапёр» идеально подходит для обучения в работе с двумерными массивами, которые моделируют игровое поле, а также с рекурсивными алгоритмами, при помощи которых раскрываются пустые клетки. Проект развивает системное мышление – от распределения памяти до логики взаимодействия элементов.

В данной работе используется язык программирования СИ, потому что этот язык обеспечивает высокую скорость выполнения, что важно для быстрой генерации поля с минами и быстрого отклика, и оптимизирует использование ресурсов при работе с динамической памятью. Также в данной работе применяется ассемблерная вставка.

1 Постановка задачи

Целью данной работы является разработка консольной версии классической игры «Сапёр» с использованием языков программирования Си и Ассемблер. В ходе выполнения работы необходимо реализовать следующий функционал:

1. Генерация игрового поля с заданными размерами и случайным размещением мин.
2. Возможность открытия клеток с использованием консольного ввода.
3. Автоматический подсчёт и отображение количества мин в соседних клетках.
4. Рекурсивное открытие смежных пустых клеток.
5. Проверка условий завершения игры.
6. Реализация системы уровней сложности с различными конфигурациями поля.

Во время реализации проекта должны быть использованы сложные типы данных – массивы, структуры, файлы, алгоритмы обработки ввода и вывода и принципы модульного программирования.

Сохранение результата игры.

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

Устройство ввода информации – клавиатура и мышь.

Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.

Наличие заставки.

2 Выбор Решения

В процессе разработки данной игры была выбрана среда разработки *Visual Studio 2022*. Она предоставляет все необходимые инструменты для комфортной и эффективной разработки игры «Сапёр», делая процесс написания кода, отладки и оптимизации максимально удобным.

Visual Studio также подходит для работы с языком ассемблера благодаря интеграции низкоуровневых возможностей с удобной средой разработки. Главным преимуществом является поддержка встроенного ассемблера, что позволяет писать код в теле программы без необходимости использования дополнительных файлов. В данной работе ассемблерная вставка демонстрирует низкоуровневый вывод текста на экран. Она используется для вывода трёх текстовых строк: название работы, название игры, информация об авторе. Вставка показывает, как реализуется вызов функции *printf* на уровне машинного кода.

Проект структурирован с использованием модулей, что позволяет логически разделить код на функциональные блоки, которые легко сопровождать. Схема данных приведена на рисунке 1.

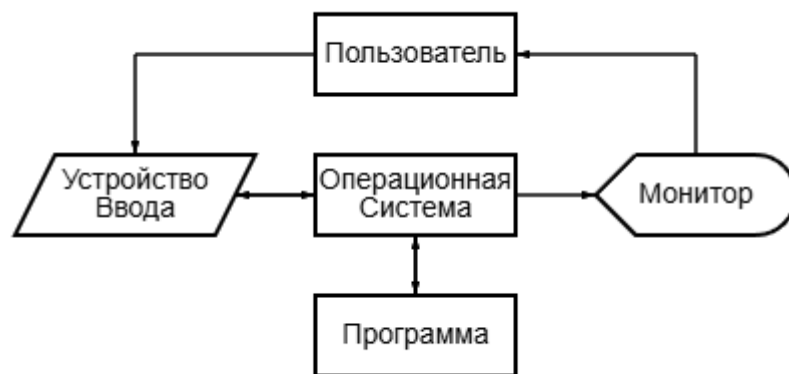


Рисунок 1 - Схема данных

Результаты игры сохраняются в текстовый файл *game_result.txt* с помощью функции *saveGameResult()*. При завершении игры в файл записываются результаты: дата, время, уровень сложности, результат игры.

3 Описание разработки программы

Для реализации был выбран нисходящий метод разработки, потому что он позволяет сначала определить общую структуру программы, а затем постепенно детализировать её компоненты.

При запуске программы на экран выводится приветственная заставка, правила пользования и 3 уровня сложности:

1. Легкий. Поле 8x8, которое содержит 10 мин;
2. Средний. Поле 12x12, которое содержит 20 мин;
3. Сложный. Поле 16x16, которое содержит 30 мин.

Программа разделена на 4 модуля, потому что это оптимальный вариант, который позволяет изменять одну из частей, не затрагивая остальные:

1. *sapper.s*;
2. *game.c*;
3. *board.c*;
4. *splash.c*.

Модуль *sapper.s* – главный файл программы, который инициализирует и запускает приложение.

Модуль *game.c* отвечает за игровую логику: выбор уровня, завершение игры и сохранение результата.

Модуль *board.c* нужен для инициализации и отображении доски, размещения мин, расчёта чисел на поле, открытия клетки и освобождения памяти.

Модуль *splash.c* нужен для отображения заставки.

4 Отладка и тестирование

В качестве среды разработки была выбрана программа *Visual Studio 2022*. Она оптимальна для разработки на языке программирования Си благодаря мощному функционалу и удобству работы.

Пошаговое выполнение дает возможность отслеживать, как программа инициализирует поле, расставляет мины и обрабатывает пользовательские ходы.

Точки останова позволяют исследовать состояние программы, с тем чтобы определить, как ведет себя приложение.

Тестирование проходило в рабочем процессе, в процессе разработки и после завершения написания программы.

На рисунке 2 показан пример использования точек останова.

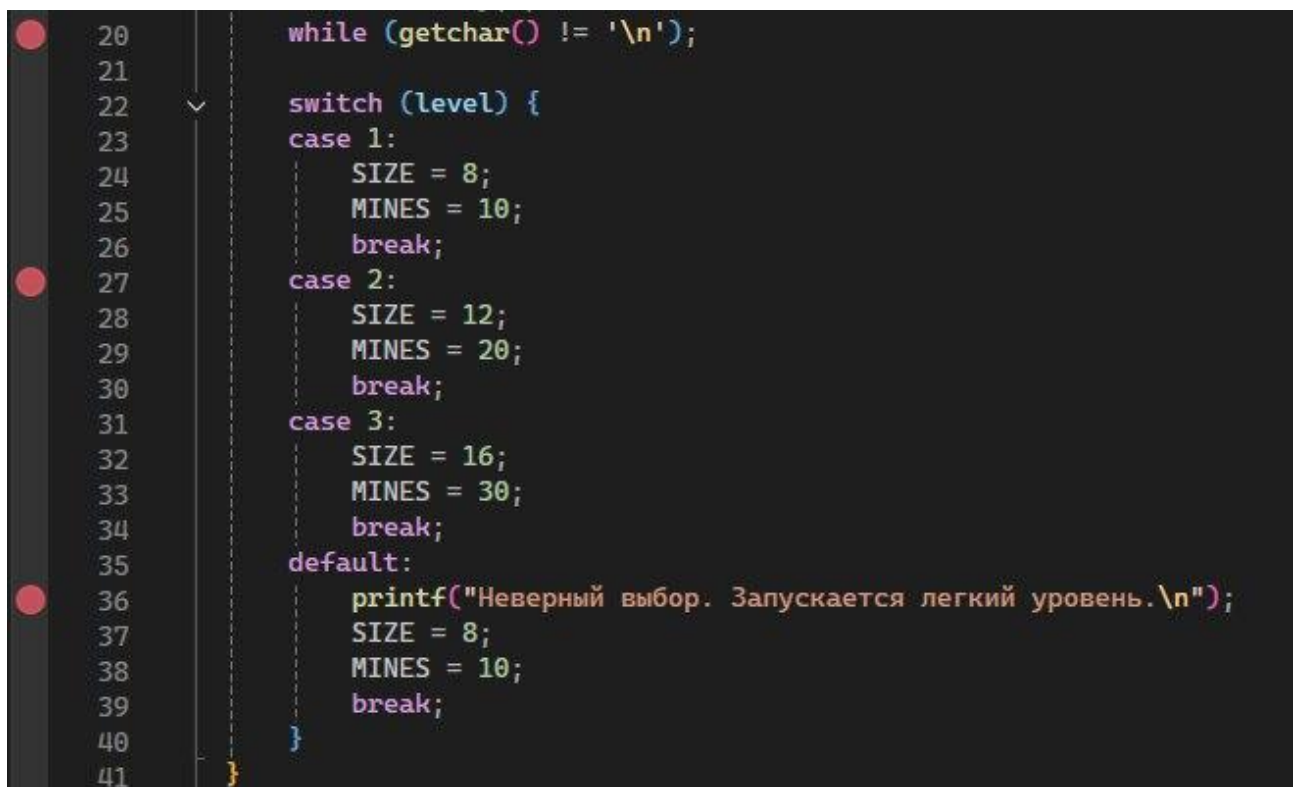


Рисунок 2 – Использование точек останова

5 Описание программы

5.1 Схема функции main

На рисунке 3 представлена схема функции main.

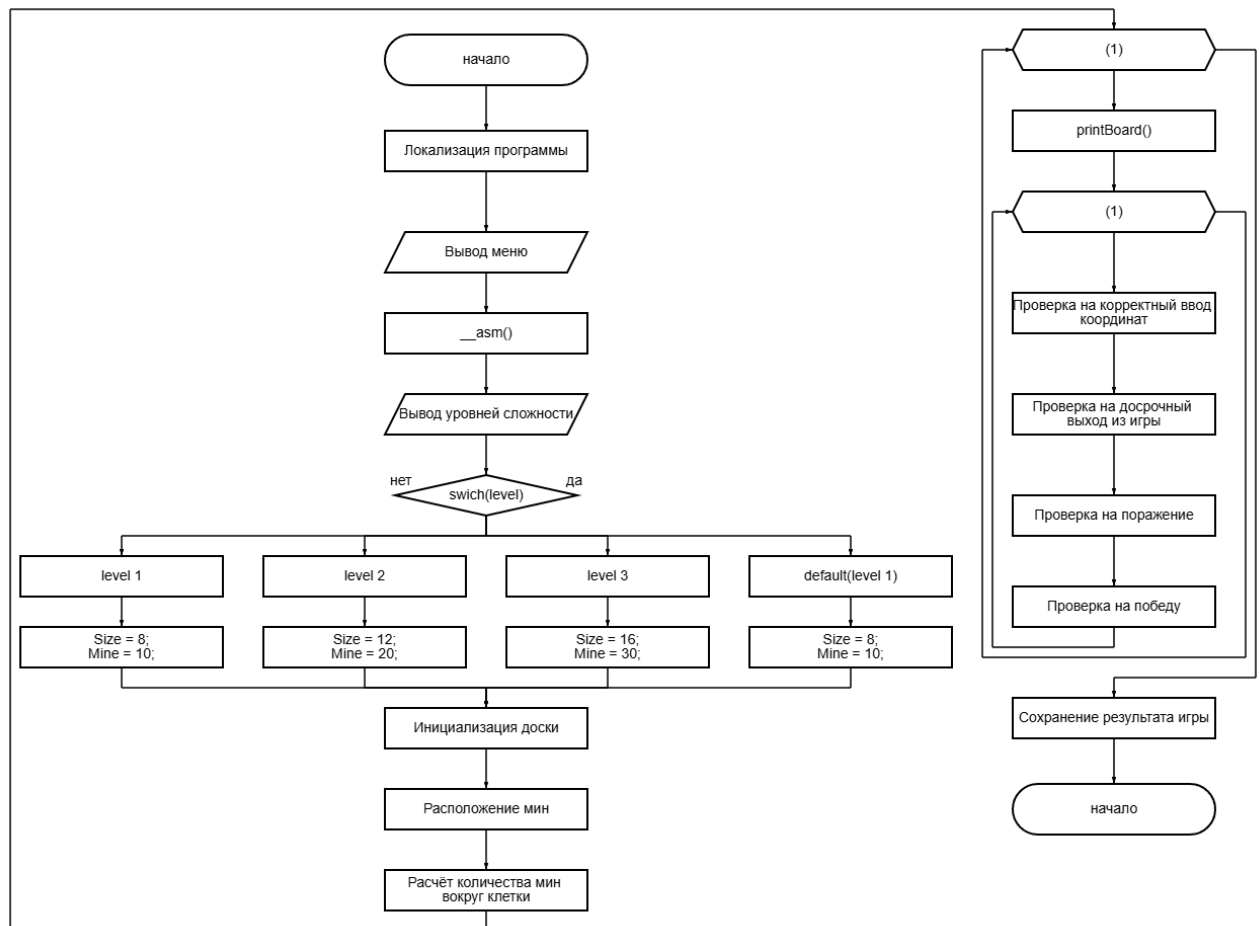


Рисунок 3 - Схема функции main

5.2 Описание модулей

Программа состоит из 4 основных модулей, которые упрощают процесс разработки и отладки, улучшают читаемость и ускоряют сборку проекта.

1) *sapper.c* – главный управляющий модуль, связывающий все компоненты программы.

1.1) *main()* – функция, в которой устанавливается русская локаль для корректного отображения кириллицы, вызываются заставка, выбор уровня сложности, инициализируется игровое поле и выполняется завершение программы.

2) *game.c* – модуль, в котором реализуются функции игровой логики.

2.1) *level()* – функция запускает уровень игры, выбранный пользователем

2.2) *saveGameResult()* – функция сохраняет результаты игры в файл

2.3) *endGame()* – функция завершает игру в случае победы, поражения или досрочного выхода

3) *game.h* – заголовочный файл, в котором объявляются функции игровой логики.

4) *board.c* – файл, в котором реализуются функции работы с игровым полем: создание, заполнение, отображение, проверка условий.

4.1) *initializeBoard()* – функция, которая инициализирует игровое поле

4.2) *placeMines()* – функция, которая размещает мины на игровом поле

4.3) *calculateNumbers()* – функция, которая производит расчёт чисел на поле

4.4) *printBoard()* – функция, которая отвечает за отображение игрового поля

4.5) *reveal()* – функция, которая отвечает за открытие клетки

4.6) *checkWin()* – функция, которая проверяет результат игры

4.7) *freeMemory()* – функция, которая освобождает память программы

5) *board.h* – заголовочный файл, в котором объявляются функции работы с файлами.

6) *splash.c* – файл, в котором выполняется функция для вывода стартового экрана.

6.1) *showSpalshScreen()* – функция, которая выводит на экран заставку игры «Сапёр».

Ассемблерная вставка находится в функции *showSplashScreen()*. Она выводит на экран тему курсовой работы и имя создателя:

```
static const char str1[] = " Курсовая работа по  
программированию\n";  
static const char str2[] = " Игра Сапёр\n";  
static const char str3[] = " Работу выполнил студент группы 24ВВВ1  
Гурин Алексей\n\n";
```

```
__asm {  
    push offset str1  
    call printf  
    add esp, 4  
  
    push offset str2  
    call printf  
    add esp, 4  
  
    push offset str3  
    call printf  
    add esp, 4  
}
```

6 Руководство пользователя

Программа sarreg.c предназначена для игры в «Сапёр». Она интуитивна понятна в управлении.

«Сапёр» – классическая игра – головоломка, где игрок открывает клетки на минном поле, стараясь избежать подрывов. Игровое поле представляет собой сетку квадратов, часть которых содержит скрытые мины. При открытии пустой клетки отображается число, указывающее количество мин в соседних клетках, что помогает игроку вычислить безопасные ходы. Если открывается клетка с миной – игра заканчивается поражением. Победа наступает, когда все безопасные клетки открыты.

6.1 Заставка

На рисунке 4 представлена игровая заставка, которая появляется при запуске приложения.

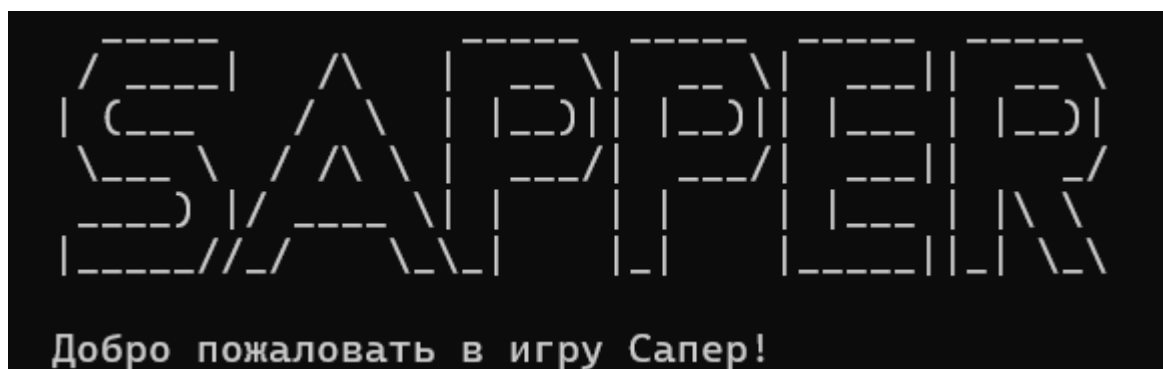


Рисунок 4 - Заставка

6.2 Правила пользования

На рисунке 5 представлены правила пользования.

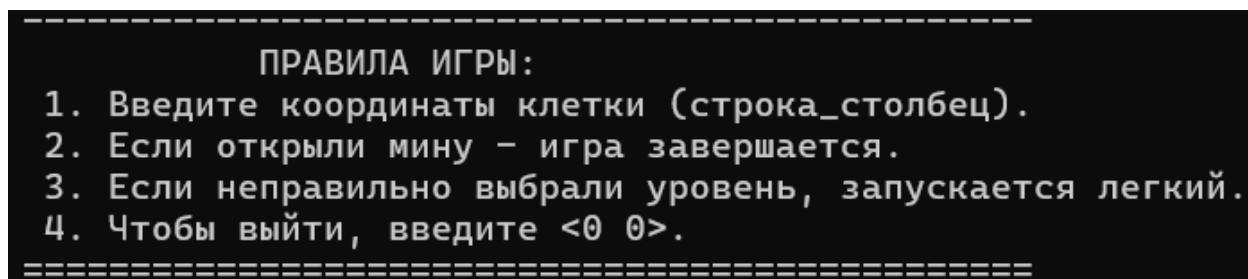


Рисунок 5 - Правила пользования

6.3 Меню уровней

При запуске приложения открывается меню уровней, в котором можно выбрать один из трёх уровней сложности: лёгкий, средний, сложный (рис. 6).

При выборе лёгкого уровня сложности, откроется игровое поле 8x8, которое содержит 10 мин.

При выборе среднего уровня сложности, откроется игровое поле 10x10, которое содержит 20 мин.

При выборе сложного уровня сложности, откроется игровое поле 16x16, которое содержит 30 мин.

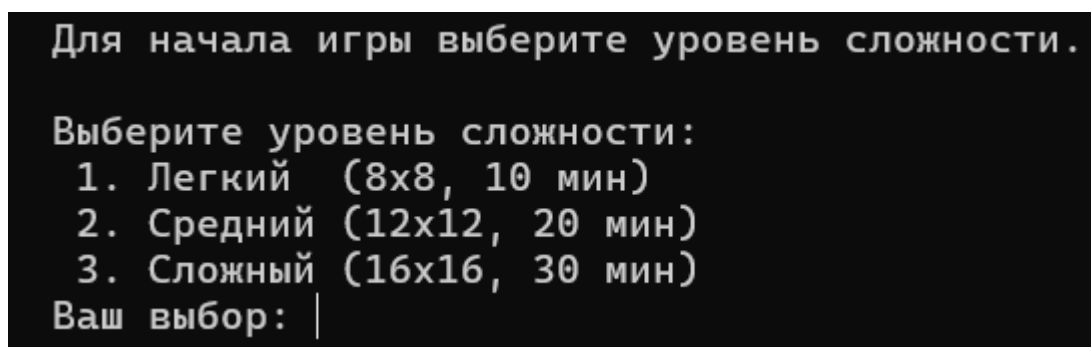


Рисунок 6 - Меню уровней сложности

6.4 Игровое поле

После выбора уровня появляется игровая доска, размер которой зависит от уровня сложности, выбранного пользователем (рис. 7).

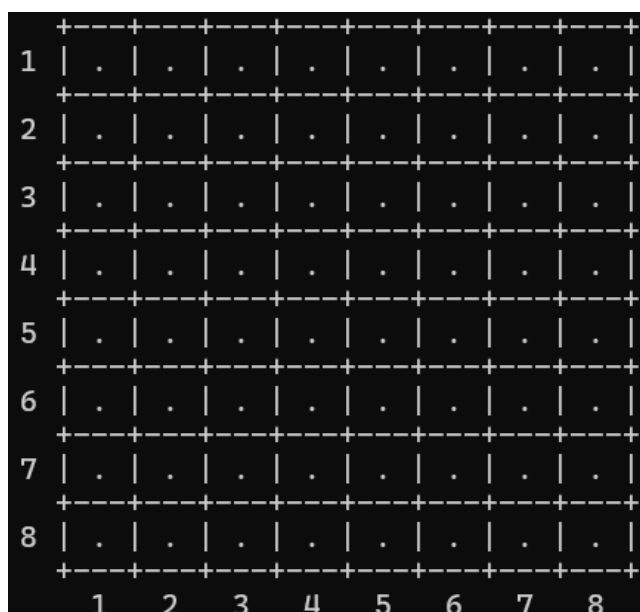


Рисунок 7 - Игровое поле

6.5 Открытие клетки

После указания координат по горизонтали и вертикали, откроется клетка на игровой доске. Если в этой клетке находится мина – игра закончится поражением. Если клетка пустая, но рядом находятся мины, то в ней появится число мин, которые находятся рядом. В противном случае откроются и выбранная, и ближайшие клетки, в которых нет мин (рис 8).

Введите координаты (строка столбец). Для выхода введите 0 0: 5 5

1		.		.		.		.		.		.		.
2		.		.		.		.		.		.		.
3		.		.		.		.		.		.		.
4		.		.		2		1		3		.		.
5		.		1		1		1		.		1		2
6		.		1		.		.		.		1		.
7		.		2		1		2		1		1		1
8		.		.		.		.		.		.		.
		1		2		3		4		5		6		7
		8												

Рисунок 8 - Открытие клетки

6.6 Аварийный выход

На рисунке 9 представлен пример аварийного выхода.

Введите координаты (строка столбец). Для выхода введите 0 0: 0 0
Игра завершена.
Результат игры сохранен в файл game_results.txt

Рисунок 9 - Аварийный выход

6.7 Сохранение результата при аварийном выводе

Если произошёл аварийный выход, в файл сохраняются дата, время, уровень сложности и результат игры – игра прервана (рис. 10).

```
-----  
Дата: 23.05.2025 11:18:54  
Уровень: 8x8, мин: 10  
Результат: Игра прервана  
-----
```

Рисунок 10 - Сохранение результатов при аварийном выходе

6.8 Поражение

Если выбрать клетку, в которой находится мина, игра завершится поражением (рис. 11).

```
Введите координаты (строка столбец). Для выхода введите 0 0: 6 5  
+---+---+---+---+---+---+---+---+  
1 |   |   |   |   |   |   | 1 | * | * |  
+---+---+---+---+---+---+---+---+  
2 | 1 | 1 | 1 |   |   |   | 1 | 2 | 2 |  
+---+---+---+---+---+---+---+---+  
3 | . | * | 1 |   |   |   |   |   |   |  
+---+---+---+---+---+---+---+---+  
4 | * | 2 | 1 |   |   |   |   |   |   |  
+---+---+---+---+---+---+---+---+  
5 | . | 1 |   |   | 1 | 2 | 2 | 1 |   |  
+---+---+---+---+---+---+---+---+  
6 | . | 1 |   |   | 1 | * | * | 2 |   |  
+---+---+---+---+---+---+---+---+  
7 | * | 2 | 1 | 2 | . | * | 3 | 1 |   |  
+---+---+---+---+---+---+---+---+  
8 | . | . | * | . | . | . | . | * |   |  
+---+---+---+---+---+---+---+---+  
    1  2  3  4  5  6  7  8  
Вы наступили на мину! Игра окончена.  
Результат игры сохранен в файл game_results.txt
```

Рисунок 11 - Поражение

6.9 Сохранение результата при поражении

Если наступить на мину, игра закончится поражением. В файл сохраняются дата, время, уровень сложности и результат игры – поражение (рис. 12).

```
-----  
Дата: 23.05.2025 09:11:03  
Уровень: 8x8, мин: 10  
Результат: Поражение  
-----
```

Рисунок 12 - Сохранение результатов при поражении

6.10 Победа

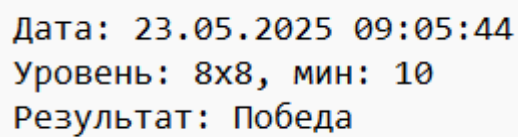
При открытии всех клеток игрового поля, в которых отсутствуют мины, игра завершится победой (рис. 13).

```
Введите координаты (строка столбец). Для выхода введите 0 0: 1 8  
+---+---+---+---+---+---+---+---+  
1 |   | 1 | . | 1 |   | 1 | 1 | 1 |  
+---+---+---+---+---+---+---+---+  
2 |   | 2 | 2 | 3 | 1 | 2 | . | 1 |  
+---+---+---+---+---+---+---+---+  
3 |   | 1 | . | 2 | . | 3 | 2 | 1 |  
+---+---+---+---+---+---+---+---+  
4 |   | 1 | 1 | 2 | 2 | . | 2 | 1 |  
+---+---+---+---+---+---+---+---+  
5 |   |   | 1 | 1 | 2 | 1 | 2 | . |  
+---+---+---+---+---+---+---+---+  
6 |   |   | 1 | . | 1 |   | 1 | 1 |  
+---+---+---+---+---+---+---+---+  
7 |   | 1 | 3 | 3 | 3 | 1 | 1 |   |  
+---+---+---+---+---+---+---+---+  
8 |   | 1 | . | . | 2 | . | 1 |   |  
+---+---+---+---+---+---+---+---+  
    1  2  3  4  5  6  7  8  
Поздравляем! Вы выиграли!  
Результат игры сохранен в файл game_results.txt
```

Рисунок 13 - Победа

6.11 Сохранение результата при победе

Если все пустые клетки игрового поля открыты, в файл сохраняются дата, время, уровень сложности и результат игры – победа (рис. 14).

A screenshot of a text file showing game results. The text is enclosed in a light gray box with dashed lines at the top and bottom. The text reads: "Дата: 23.05.2025 09:05:44", "Уровень: 8x8, мин: 10", and "Результат: Победа".

```
-----  
Дата: 23.05.2025 09:05:44  
Уровень: 8x8, мин: 10  
Результат: Победа  
-----
```

Рисунок 14 - Сохранение результатов при победе

Результат игры сохраняется в *файл game_results.txt*.

Заключение

В ходе выполнения курсовой работы была разработана программа - компьютерная игра «Сапёр», реализованная на языках программирования Си и Ассемблер. Данный проект позволил закрепить и углубить ключевые навыки системного программирования, а также продемонстрировал эффективное применение возможностей языка Си для решения практических задач.

Основные достижения и реализованные функции:

1. Была успешно применена работа с динамической памятью, включая:
 - создание и освобождение двумерных массивов;
 - оптимизацию использования памяти;
 - предотвращение утечек памяти.
2. реализована сложная логика игры, включающая:
 - генерацию случайного расположения мин;
 - рекурсивный алгоритм открытия клеток;
 - механизм подсчета соседних мин;
 - систему обработки пользовательского ввода.
3. применены низкоуровневые оптимизации:
 - использование ассемблерных вставок;
 - оптимизация работы с памятью;
 - улучшение производительности алгоритмов.
4. разработан удобный пользовательский интерфейс:
 - система выбора уровня сложности;
 - наглядное отображение игрового поля.

Выполненная работа продемонстрировала эффективность языка Си для разработки сложных логических игр. Программа успешно реализует все основные функции классической игры "Сапёр", демонстрируя при этом высокую производительность и надежность. Приобретенные в ходе работы навыки имеют важное значение для дальнейшего профессионального роста в области системного программирования.

Список используемых материалов

1. CppReference - <https://en.cppreference.com/w/c>
2. Microsoft C/C++ Documentation - <https://docs.microsoft.com/cpp/c-language/>
3. GeeksforGeeks C Programming - <https://www.geeksforgeeks.org/c-programming-language/>
4. Stack Overflow - <https://stackoverflow.com/questions/tagged/c>
5. GCC Inline Assembly HOWTO: <https://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html>
6. C Programming Subreddit: https://reddit.com/r/C_Programming

Приложение А

Листинг программы

Файл sapper.c

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>
#include "splash.h"
#include "board.h"
#include "game.h"

int main() {
    setlocale(LC_ALL, "Rus");
    srand(time(NULL));

    showSplashScreen();
    level();
    initializeBoard();
    placeMines();
    calculateNumbers();
    endGame();
    freeMemory();
    return 0;
}
```

Файл splash.c

```
#define _CRT_SECURE_NO_WARNINGS
#include "splash.h"

void showSplashScreen() {
    static const char str1[] = " Курсовая работа по
программированию\n";
    static const char str2[] = " Игра Сапёр\n";
    static const char str3[] = " Работу выполнил студент группы
24BBB1 Гурин Алексей\n\n";

    __asm {
        push offset str1
        call printf
        add esp, 4

        push offset str2
        call printf
        add esp, 4

        push offset str3
        call printf
        add esp, 4
    }
    printf("      _____\n");
    printf(" / _____ /\\ | _ \\| _ \\| _|| _ \\|\\n");
    printf(" | (_____/ \\ | |_)|| |_)|| |__ | |_)|\\n");
    printf(" \\ \\__ \\ / \\ \\ \\ | __/| __/| __|| _/\\n");
    printf(" ____ ) |/_ \\ \\| | | | |__ | |\\ \\ \\|\\n");
    printf(" |____//_/_ \\ \\_\\_| | _ |____||_| \\ \\_\\|\\n");
    printf("\\n");
    printf(" Добро пожаловать в игру Сапер!\\n");
    printf("-----\\n");
    printf("          ПРАВИЛА ИГРЫ:          \\n");
    printf(" 1. Введите координаты клетки (строка_столбец).\\n");
    printf(" 2. Если открыли мину – игра завершается.          \\n");
```

```

    printf(" 3. Если неправильно выбрали уровень, запускается
легкий.\n");
    printf(" 4. Чтобы выйти, введите <0 0>.                \n");
    printf("=====\n");
    printf("\n");
    printf(" Для начала игры выберите уровень сложности.\n");
    printf("\n");
}

```

Файл splash.h

```

#ifndef SPLASH_H
#define SPLASH_H
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
#include <string.h>
void showSplashScreen();

#endif

```

Файл board.c

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include "board.h"

int SIZE;
int MINES;
int MOVES = 0;
int** board;
int** revealed;

void initializeBoard() {
    board = (int**)malloc(SIZE * sizeof(int*));
    revealed = (int**)malloc(SIZE * sizeof(int*));
    if (!board || !revealed) {

```

```

        printf("Ошибка выделения памяти!\n");
        exit(1);
    }

    for (int i = 0; i < SIZE; i++) {
        board[i] = (int*)malloc(SIZE * sizeof(int));
        revealed[i] = (int*)malloc(SIZE * sizeof(int));
        if (!board[i] || !revealed[i]) {
            printf("Ошибка выделения памяти!\n");
            exit(1);
        }
        for (int j = 0; j < SIZE; j++) {
            board[i][j] = 0;
            revealed[i][j] = 0;
        }
    }
}

void placeMines() {
    int mines_placed = 0;
    while (mines_placed < MINES) {
        int x = rand() % SIZE;
        int y = rand() % SIZE;

        if (board[x][y] != -1) {
            board[x][y] = -1;
            mines_placed++;
        }
    }
}

void calculateNumbers() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (board[i][j] == -1) continue;
            int count = 0;

```

```

        for (int x = -1; x <= 1; x++) {
            for (int y = -1; y <= 1; y++) {
                if (i + x >= 0 && i + x < SIZE && j + y >= 0
&& j + y < SIZE && board[i + x][j + y] == -1) {
                    count++;
                }
            }
        }
        board[i][j] = count;
    }
}

void printBoard() {
    printf("  +");
    for (int i = 0; i < SIZE; i++) {
        printf("---+");
    }
    printf("\n");

    for (int i = 0; i < SIZE; i++) {
        printf("%2d |", i + 1);
        for (int j = 0; j < SIZE; j++) {
            if (revealed[i][j]) {
                if (board[i][j] == -1) {
                    printf(" * |");
                }
                else if (board[i][j] == 0) {
                    printf("  |");
                }
                else {
                    printf("%2d |", board[i][j]);
                }
            }
            else {
                printf(" . |");
            }
        }
    }
}

```

```

    }
    printf("\n");

    printf("    +");
    for (int j = 0; j < SIZE; j++) {
        printf("---+");
    }
    printf("\n");
}

printf("    ");
for (int i = 1; i <= SIZE; i++) {
    printf("%2d  ", i);
}
printf("\n");
}

int reveal(int x, int y) {
    if (x < 1 || x > SIZE || y < 1 || y > SIZE || revealed[x -
1][y - 1]) {
        return 0;
    }
    revealed[x - 1][y - 1] = 1;
    MOVES++; // Увеличиваем счетчик ходов
    if (board[x - 1][y - 1] == -1) {
        return -1;
    }
    if (board[x - 1][y - 1] == 0) {
        for (int i = -1; i <= 1; i++) {
            for (int j = -1; j <= 1; j++) {
                if (x + i >= 1 && x + i <= SIZE && y + j >= 1 && y
+ j <= SIZE) {
                    reveal(x + i, y + j);
                }
            }
        }
    }
}

```

```

    }
    return 0;
}

int checkWin() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (board[i][j] != -1 && !revealed[i][j]) {
                return 0;
            }
        }
    }
    return 1;
}

void freeMemory() {
    for (int i = 0; i < SIZE; i++) {
        free(board[i]);
        free(revealed[i]);
    }
    free(board);
    free(revealed);
}

```

Файл board.h

```

#ifndef BOARD_H
#define BOARD_H

extern int SIZE;
extern int MINES;
extern int MOVES;
extern int** board;
extern int** revealed;

void initializeBoard();
void placeMines();
void calculateNumbers();

```

```
void printBoard();
int reveal(int x, int y);
int checkWin();
void freeMemory();
```

```
#endif
```

Файл game.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "board.h"
#include "game.h"

void level() {
    int level;
    printf(" Выберите уровень сложности:\n");
    printf("  1. Легкий   (8x8, 10 мин)\n");
    printf("  2. Средний  (12x12, 20 мин)\n");
    printf("  3. Сложный  (16x16, 30 мин)\n");
    printf(" Ваш выбор: ");
    scanf("%d", &level);

    while (getchar() != '\n');

    switch (level) {
    case 1:
        SIZE = 8;
        MINES = 10;
        break;
    case 2:
        SIZE = 12;
        MINES = 20;
        break;
    case 3:
```

```

        SIZE = 16;
        MINES = 30;
        break;
default:
    printf("Неверный выбор. Запускается легкий уровень.\n");
    SIZE = 8;
    MINES = 10;
    break;
}
}

void saveGameResult(const char* result) {
    FILE* file = fopen("game_results.txt", "a");
    if (file == NULL) {
        printf("Ошибка открытия файла для сохранения
результатов.\n");
        return;
    }

    time_t now;
    time(&now);
    struct tm* local = localtime(&now);

    fprintf(file, "Дата: %02d.%02d.%04d %02d:%02d:%02d\n",
        local->tm_mday, local->tm_mon + 1, local->tm_year + 1900,
        local->tm_hour, local->tm_min, local->tm_sec);
    fprintf(file, "Уровень: %dx%d, мин: %d\n", SIZE, SIZE, MINES);
    fprintf(file, "Результат: %s\n", result);
    fprintf(file, "-----\n");
    fclose(file);

    printf("Результат игры сохранен в файл game_results.txt\n");
}

void endGame() {
    int x, y;
    char input[256];

```

```

char* endptr;
long val_x, val_y;

while (1) {
    printBoard();

    while (1) {
        printf("Введите координаты (строка столбец). Для
выхода введите 0 0: ");

        if (fgets(input, sizeof(input), stdin) == NULL) {
            printf("Ошибка ввода.\n");
            continue;
        }

        input[strcspn(input, "\n")] = '\0';

        if (strcmp(input, "0 0") == 0) {
            x = 0; y = 0;
            break;
        }

        val_x = strtol(input, &endptr, 10);
        if (endptr == input || (*endptr != ' ' && *endptr !=
'\t')) {
            printf("Некорректный ввод.\n");
            continue;
        }

        while (*endptr == ' ' || *endptr == '\t') endptr++;

        val_y = strtol(endptr, &endptr, 10);
        if (*endptr != '\0') {
            printf("Некорректный ввод.\n");
            continue;
        }
    }
}

```

```

        x = (int)val_x;
        y = (int)val_y;
        break;
    }
    if (x == 0 && y == 0) {
        printf("Игра завершена.\n");
        saveGameResult("Игра прервана");
        break;
    }
    if (reveal(x, y) == -1) {
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                if (board[i][j] == -1) revealed[i][j] = 1;
            }
        }
        printBoard();
        printf("Вы наступили на мину! Игра окончена.\n");
        saveGameResult("Поражение");
        break;
    }
    if (checkWin()) {
        printBoard();
        printf("Поздравляем! Вы выиграли!\n");
        saveGameResult("Победа");
        break;
    }
}
}

```

Файл game.h

```

#ifndef GAME_H
#define GAME_H

void level();
void saveGameResult(const char* result);
void endGame();

#endif

```