

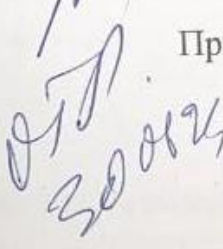
Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по курсу «Программирование»
на тему «Разработка многомодульной программы
«Игра «Морской бой»»»



Выполнил:

студент группы 23BBB2
Чупраков С. В.



Принял:

Слепцов Н.В.

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 Чупракову Сергею Владимировичу Группа 23 ВВВ2

Тема проекта: **Игра «Морской бой»**

Исходные данные (технические требования) на проектирование

1. *Предусмотреть графический интерфейс*
2. *Несколько уровней игры, отличающихся по размеру поля, количеству и способности размещения игроков объектов (минные корабли и корабли)*
3. *Предусмотреть ведение в истории игры 10 лучших игроков*

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 " 6 " февраля 2024 г.

Дата защиты проекта " " "

Руководитель _____

Задание получил " 25 " марта 2024 г.

Студент Гурганов Сергей Владимирович

С.Г. Гурганов

Оглавление

Введение	3
1. Постановка задачи	4
2. Выбор решения	5
2.1 Определение необходимых модулей программы.....	5
2.2 Определение необходимых структур данных	5
3. Описание разработки программы	7
4. Описание программы.....	9
4.1 Разработка меню	9
4.2 Разработка уровней игры	10
4.3 Разработка функции на языке Assembler	13
5. Отладка и тестирование.....	15
6. Руководство пользователя	17
Приложение А Листинги программы	21
Приложение А.1 Файл «battleship.cpp».....	21
Приложение А.2 Файл «cur.h».....	23
Приложение А.3 Файл «asmpart.cpp».....	24
Приложение А.4 Файл «10.cpp»	25
Приложение А.5 Файл «11.cpp».....	36
Приложение А.6 Файл «12.cpp»	48
Приложение В Снимки экрана	59
Приложение С Результат работы программы	61
Приложение С.1 Файл s.txt	61

Введение

В современном мире компьютерные игры стали неотъемлемой частью нашей жизни. Они позволяют нам отдохнуть, развлечься, соревноваться с друзьями, а иногда даже обучаться. Их создание было бы невозможным без программирования.

Язык программирования Си - это универсальный язык с богатым набором операторов и компактным способом записи выражений. Благодаря гибкости, выразительности и компактности своих конструкций. Си завоевал наибольшую популярность в среде профессиональных программистов и широко используется при разработке системных и прикладных программ.

В данной работе продемонстрированы базовые возможности языка СИ вкпе с ассемблерными вставками. Разработка велась в среде Microsoft Visual Studio 2019.

Разработка компьютерной игры Морской бой до сих пор остается актуальной и востребованной среди игроков всех возрастов. Создание данной игры требует знания различных функций языка программирования, что является хорошей тренировочной платформой для начинающих разработчиков. Поэтому разработка новых версий и вариаций игры Морской бой остается актуальной и важной задачей.

1. Постановка задачи

Необходимо разработать программу «Игра «Морской бой»». Цель игры – уничтожить корабли противника, расположенные на игровом поле, выбирая координаты ударов по врагу.

В игре должно присутствовать несколько уровней сложности, в которых отличается размер поля, ведение игрового счёта. Для этого необходимо определить их через сложные типы данных.

Также необходимо сохранять и отображать 10 лучших результатов. Для реализации этого пункта необходимо определить структуру базы данных. Изучить функции для работы с файлами.

Необходима Заставка-приветствие, которая должна сообщать пользователю необходимую и достаточную информацию о программе. Заставка сообщает пользователю информацию о программе.

Также программа должна отвечать следующим требованиям:

1. Многомодульность. Программа должна быть поделена на логические модули.

2. Использование сложных типов данных: массивов, структур и т.д. Для разработки данной игры необходимо подробно изучить принципы работы с массивами и структурами для создания игрового поля и реализации графики.

3. Режим работы видеосистемы: текстовый/графический. Необходимо определиться с типом управления, видом интерфейса и изучить приёмы реализации графики.

4. Устройство ввода информации: клавиатура/мышь. Необходимо определиться с механикой взаимодействия игрока с игрой и её интерфейсом.

5. Пользовательский интерфейс должен быть построен на основе меню или панели инструментов. Необходимо реализовать пользовательское меню.

6. Операционная система: MS Windows. При разработке использовать среду MS Windows с системой программирования MS Visual Studio.

7. Язык программирования: Си и Ассемблер, а также некоторые функции из языка C++.

2. Выбор решения

2.1 Определение необходимых модулей программы

Разработанная программа состоит из четырех модулей.

1. Файл battleship.cpp, заголовочный файл cur.h и asmpart.cpp
2. Файл 10.cpp
3. Файл 11.cpp
4. Файл 12.cpp

Заголовочный файл cur.h используются для подключения необходимых библиотек и содержат описание структур и константных значений. Данный файл необходим для корректной работы почти всех файлов. В заголовочном файле cur.h объявляются функции-уровни.

Модуль battleship.cpp является основным для данной курсовой работы. Модуль использует ещё три модуля 10.cpp, 11.cpp и 12.cpp, которые являются тремя размерами игрового поля. Тем самым удовлетворяется требование многомодульности данной курсовой работы.

Управление приложением осуществляется с помощью клавиатуры. При запуске приложения выводится заставка-приветствие, в котором описаны цель и правила игры, а также дана справка по управлению, после чего программа готова к работе. В меню игрок вводит свое имя и выбирает один из трех уровней сложности. Уровни отличаются размером игрового поля. Игрок выбирает координаты удара по полю врага, если удар был нанесен в море, ход переходит к компьютеру. Управление реализовано через ввод координат с клавиатуры. Уничтожив все корабли противника, игрок побеждает. В случае проигрыша он возвращается в меню. В меню можно вернуться и по нажатию клавиши Esc. За разрушение блоков игроку начисляются очки. Если игрок наберет достаточно много очков, после окончания игры его имя и счет будут занесены в список 10 лучших результатов.

2.2 Определение необходимых структур данных

Проанализировав цели и принципы игры, были разработаны следующие структуры.

Структура `TPlayer` определяет имя и счет игроков в таблице лидеров. Первый параметр содержит данные типа `char[5]` и хранят имя игрока. Второй параметр типа `int` хранит счет.

```
typedef struct {  
    char name[5];  
    int score;  
}TPlayer;
```

3. Описание разработки программы

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. При разработке языка Си был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. В языке Си предусмотрено много операций, непосредственно исполняемых большинством микропроцессоров; в то же время он дает программисту возможность выражать свои алгоритмы и данные наиболее подходящими средствами. Язык Си обеспечивает возможности структурирования данных. Он позволяет разрабатывать большие, но структурированные программы, предоставляя возможность отдельной разработки подпрограмм.

Целью создателей языка Си была разработка мобильного языка, который можно было бы использовать для разработки системного программного обеспечения. Программы, написанные на языке высокого уровня, мобильны, поскольку язык должен быть одним и тем же независимо от того, на каком компьютере и в какой операционной системе он используется. Дополнительное свойство языка Си, называемое условной компиляцией, позволяет программисту изолировать машинно-зависимые операторы и контролировать их компиляцию в другой среде. Это дополнительно повышает мобильность программного обеспечения, написанного на языке Си.

В качестве среды программирования был выбран программный продукт VisualStudio 2019.

Морской бой - это классическая настольная игра для двух игроков, целью которой является потопить все корабли противника. Игра состоит из игрового поля, на котором расположены корабли в виде прямоугольных клеток: типы - однопалубные, двухпалубные, трехпалубные и четырехпалубные корабли. В программе игровые поля реализованы в виде двумерных массивов. Размер кораблей тоже задается через массив.

Игроки по очереди называют координаты клеток на игровом поле противника, пытаясь угадать местоположение его кораблей. Если выстрел игрока попадает в клетку с кораблем, он объявляет: "Попал!" и отмечает эту

клетку на своем игровом поле. Если же игрок не попадает, он объявляет: "Промех!" и отмечает клетку, в которую стрелял.

Взаимодействие компьютера и игрока реализовано через функции, в которых вводятся координаты клеток для ударов по клеткам противника.

Цель игры - первым потопить все корабли противника, попадая в все клетки, занимаемые кораблями. Побеждает игрок, который первым разгадает местоположение всех кораблей противника.

Также в игре существует такой объект, как мины. При попадании по мине, игрок или компьютер теряют одну палубу своего корабля.

Интерфейс игры будет представлен в консоли. Игроку необходимо составить свое поле отдельно на листе бумаги и в компьютер оно не вводится. Ход совершается путем ввода координат поля и ответа на вопросы бота.

При запуске программы пользователь увидит приветствие, а ниже справку об игре и инструкцию по управлению. Далее пользователь должен заполнить два поля. Первый – имя, под которым будет записан его рекорд в случае попадания его в список лучших результатов. Второй – выбор уровня сложности. Это реализовано с помощью ввода чисел соответствующих размерам поля от 10x10 до 12x12 клеток. Они вызывают один из модулей уровня Файл 10.cpp, Файл 11.cpp или Файл 12.cpp.

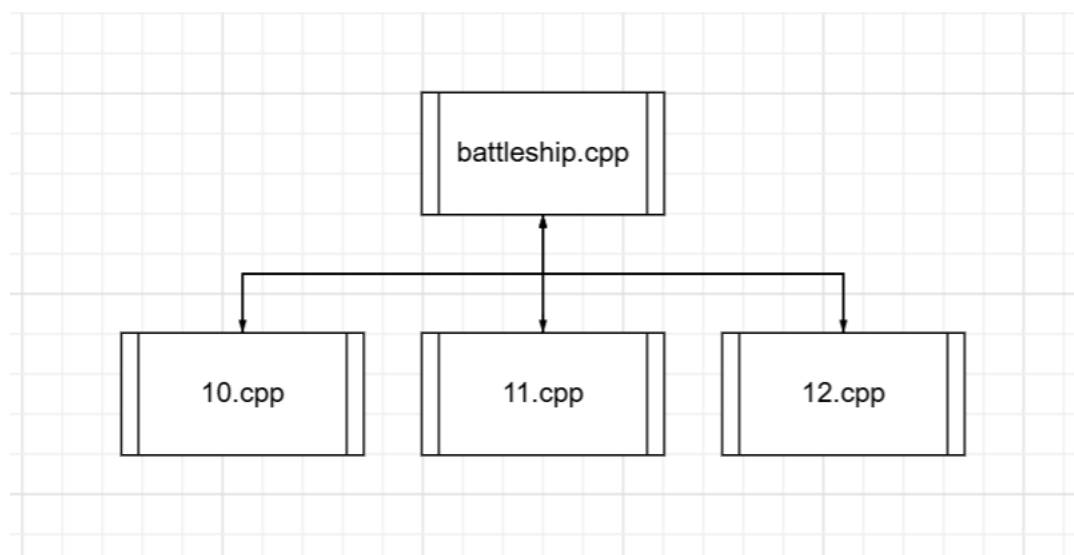


Рисунок 1 – Схема связи модулей программы

4. Описание программы

4.1 Разработка меню

Основным модулем является battleship.cpp. В нем при помощи стандартных функций ввода и вывода реализован интерфейс программы. При запуске программы выводится картинка-заставка, приветствие, справка об игре и информация об управлении. Затем в цикле находятся поля для ввода имени и выбора сложности. Описание всех состояний выполнено в таблице 1.

Таблица 1 – Описание состояний программы

Ключевое слово, вызывающее событие	Действие пользователя	Действия программы
10	Введено ключевое слово 10	Вызван модуль 10
11	Введено ключевое слово 11	Вызван модуль 11
12	Введено ключевое слово 12	Вызван модуль 12
exit	Введено ключевое слово exit	Выход из программы

Схема модуля battleship.cpp представлен на рисунке 2.

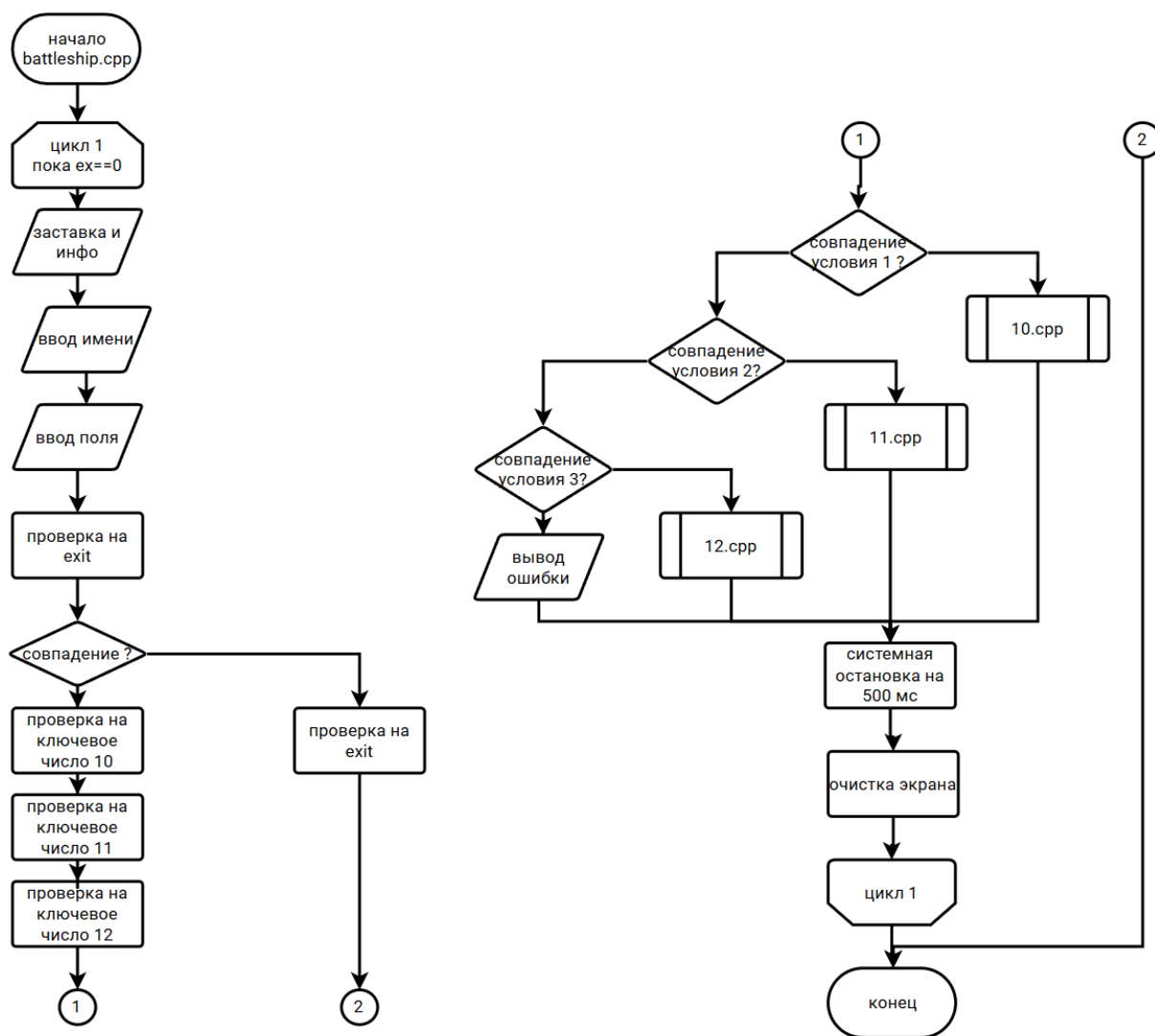


Рисунок 2 – Схема модуля battleship.cpp

4.2 Разработка уровней игры

Модули 10.cpp, 11.cpp и 12.cpp можно считать ключевыми в проекте и их необходимо разобрать также как основной модуль. Модуль 10.cpp является шаблоном для модулей 11.cpp и 12.cpp и их алгоритмы практически идентичны. Отличие заключается в некоторых отдельных переменных и функциях, которые однако отличаются только значением и названием. Поэтому будет разобран только модуль 10.cpp.

В заголовочном файле sig.h подключены библиотеки, используемые в проекте. В начале модуля 10.cpp объявлены массивы `a[10][10]`, `z[10][10]`. Это массивы содержащие информацию об игровых полях компьютера и игрока.

Объявляются массивы `b[20][3]`, `mz[10][2]`, `c[10][3]`, `m[3][3]`. Массив `b` содержит информацию о координатах кораблей компьютера. Массив `mz` содержит информацию о координатах подбитых палуб кораблей для отслеживания ботом в ходе игры. Массив `c` содержит информацию о размерах кораблей заданных перед началом игры. Массив `m` содержит информацию о количестве мин заданных перед началом игры.

Объявляется переменная `size`, которая используется в функциях для работы с размерами поля. Инициализируется переменная `num_of_hits1`, которая содержит количество выстрелов, сделанных игроком. Для того чтобы в конце подсчитать итоговый счет, инициализируется указатель на данную переменную `Phits`. Инициализируются указатели на массивы `a`, `z` и `b`. Инициализируются функции, используемые в главной функции `index10()`. Инициализируется функция `SpawnOneShip1()`, которая расставляет корабли на поле, следующая функция `SpawnOneMina1()`, которая расставляет мины на поле. Функция `input1()` вызывает функции `SpawnOneShip1`. и `SpawnOneMina1()` по количеству элементов, заданных в массивах `c` и `m`. Функция `proverka1()`, которая проверяет карту на наличие уничтоженных кораблей и отмечает их на поле. Функции `human1()` отвечает за ход игрока и указание попаданий игрока по врагу. Функция `mind1()`, которая является функцией компьютера и в ней содержится алгоритм бота, координирующего удары по полю игрока. Данные функции используются в главной функции этого модуля `index10()`. Основные действия программы реализованы с помощью цикла, который повторяется до тех пор, пока количество подбитых кораблей человеком у бота или количество подбитых кораблей ботом у человека не превысит количество кораблей, заданное в начале игры. Цикл начинается с выполнения функции `human1()`. Если игрок попал, то функция вызывается снова. В случае промаха ход переходит к компьютеру в функции `mind1()`, функция принимает ответы игрока в виде слов, которые влияют на дальнейшие действия компьютера. В функции `human1()` существует возможность написать "стоп" для выхода в меню, в этом случае цикл прервется и на экран будет выведена надпись "Выход в меню". В случае, когда количество кораблей, пораженных компьютером, соответствует общему количеству кораблей, игра заканчивается, выводится надпись о поражении игрока и само поле игрока. В ином случае выводится надпись о

победе игрока, подсчитывается счет, ставится флаг win и функция возвращается в меню, где в случае победы и получения достаточно высокого счета результат заносится в таблицу лидеров.

Схема модуля 10.cpp представлен на рисунке 3.

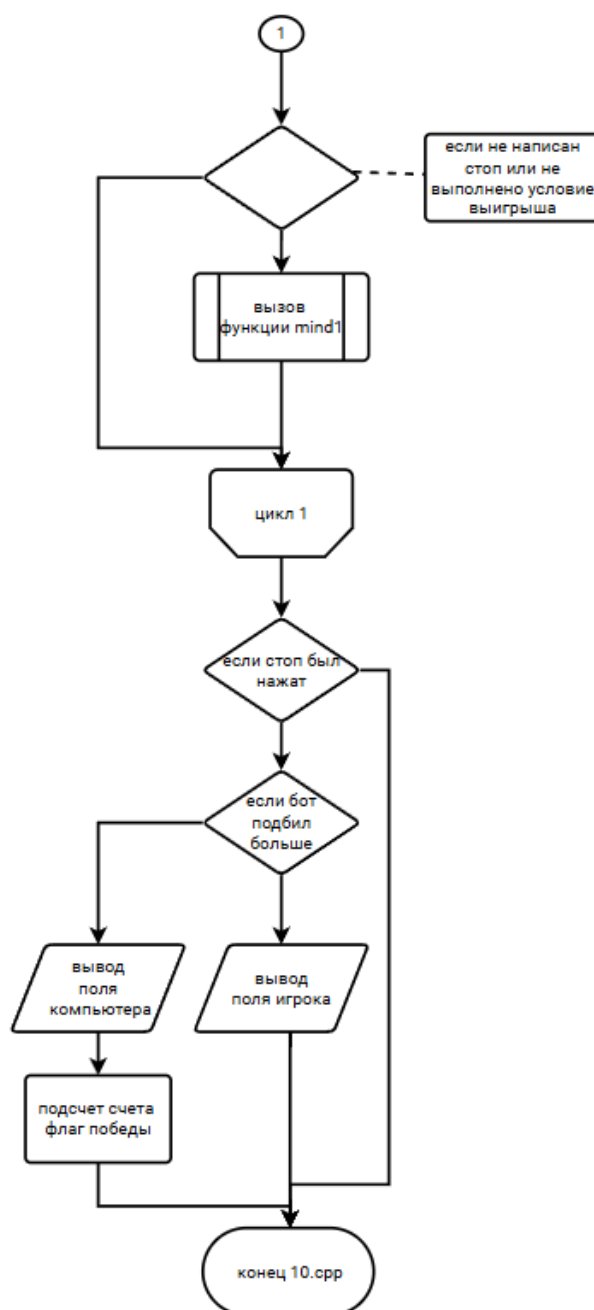


Рисунок 3 – Схема модуля 10.cpp

4.3 Разработка функции на языке Assembler

В качестве разрабатываемой функции была выбрана функция `pr()`, которая выводит на экран игровое поле, преобразуя числа массива в символы.

Код функции также представлен в приложении.

Описание функции выполнено на языке Си, а тело функции представляет собой ассемблерную вставку. Компилятор Visual Studio 2019, позволяет это делать. Для этого перед вставкой на языке Assembler необходимо написать ключевое слово “`__asm`”, а сам код поместить в фигурные скобки.

```
#include "cur.h"

void pr(int* a, int size) {

    if (size == 10){
        printf("    А В В Г Д Е Ё Ж З И\n");
    }
    else if(size == 11){
        printf("    А В В Г Д Е Ё Ж З И К\n");
    }
    else if(size == 12){
        printf("    А В В Г Д Е Ё Ж З И К Л\n");
    }

    for (int i = 0; i < size; i++) {
        printf(" %d", i + 1);
        if (i < 9) {
            printf(" ");
        }
        for (int j = 0; j < size; j++) {
            char p1[] = " _", p2[] = " O", p3[] = " M", p4[] = " X";
            int yy = *(a + i*size + j);
            __asm {
                mov eax, yy
                cmp eax, 0
                JNE skipl11
                lea edx, p1;
                push edx;
                call printf
                pop edx
                jmp skip
                skipl11 :

                cmp eax, 5
                JNE skipl1
                lea edx, p1;
                push edx;
                call printf
                pop edx
                jmp skip
                skipl1 :

                cmp eax, 8
                JNE skip1
                lea edx, p1;
                push edx;
```

```

        call printf
        pop edx
        jmp skip
    skip1 :

        cmp eax, 9
        JNE skip4
        lea edx, p1;
    push edx;
    call printf
    pop edx
    jmp skip
    skip4 :

    mov eax, yy
    cmp eax, 3
    JNE skip2
    lea edx, p2;
    push edx;
    call printf
    pop edx
    jmp skip
    skip2 :

    mov eax, yy
    cmp eax, 4
    JNE skip3
    lea edx, p3;
    push edx;
    call printf
    pop edx
    jmp skip
    skip3 :

    mov eax, yy
    cmp eax, 6
    JNE skip
    lea edx, p4;
    push edx;
    call printf
    pop edx

    skip :
    }
}
printf("\n");
}

```

Для реализации функции `pr()` были использованы следующие ассемблерные инструкции:

- инструкция `push` выполняет “вталкивание” используемых регистров в стек
- инструкция `mov` выполняет копирование содержимого источника в приемник
- инструкция `cmp` сравнивает число в массиве с числом соответствующим выводу символа
- инструкция `lea` загружает адрес символа для вывода на экран

-инструкция pop выполняет выталкивание используемых регистров из стека.

5. Отладка и тестирование

В качестве среды разработки была выбрана программа Visual Studio 2019. Программа предоставляет все средства, необходимые при разработке и отладке модулей и программ. В ходе отладки применялись такие возможности, как точка останова, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти.

Отладчик — это узкоспециализированное средство разработки, которое присоединяется к работающему приложению и позволяет проверять код. В документации по отладке для Visual Studio именно это обычно подразумевается под отладкой. Отладчик является важнейшим инструментом для поиска и устранения ошибок в приложениях.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы.

В ходе отладки была выявлена проблема с вводом букв в координатах полей размером 11 и 12 клеток.

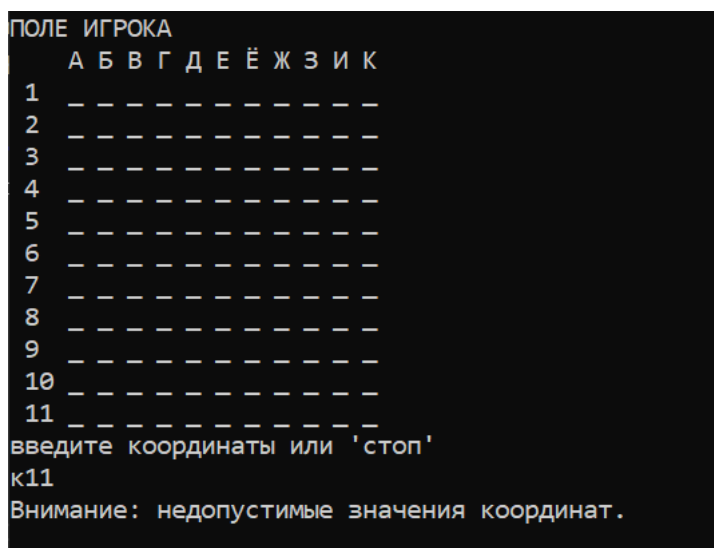


Рисунок 4 – Ошибка при вводе координат

После изменения кода программа успешно приняла значение координат, что было отражено на поле компьютера.

```
ПОЛЕ КОМПЬЮТЕРА
  А Б В Г Д Е Ё Ж З И К
1  - - - - - - - - - -
2  - - - - - - - - - -
3  - - - - - - - - - -
4  - - - - - - - - - -
5  - - - - - - - - - -
6  - - - - - - - - - -
7  - - - - - - - - - -
8  - - - - - - - - - -
9  - - - - - - - - - -
10 - - - - - - - - - -
11 - - - - - - - - - О

ПОЛЕ ИГРОКА
  А Б В Г Д Е Ё Ж З И К
1  - - - - - - - - - -
2  - - - - - - - - - -
3  - - - - - - - - - -
4  - - - - - - - - - -
5  - - - - - - - - - -
6  - - - - - - - - - -
7  - - - - - - - - - -
8  - - - - - - - - - -
9  - - - - - - - - - -
10 - - - - - - - - - -
11 - - - - - - - - - -
Captain PC: Г 2
Введите 'промах' 'попал' 'убил' 'мина'
```

Рисунок 5 – Решение ошибки при вводе координат

6. Руководство пользователя

Программа battleship.cpp реализует игру Морской бой, цель которой уничтожить все корабли противника. Программа имеет простое меню для ввода имени, перехода к игровым уровням и выхода из программы с помощью вводов с клавиатуры.

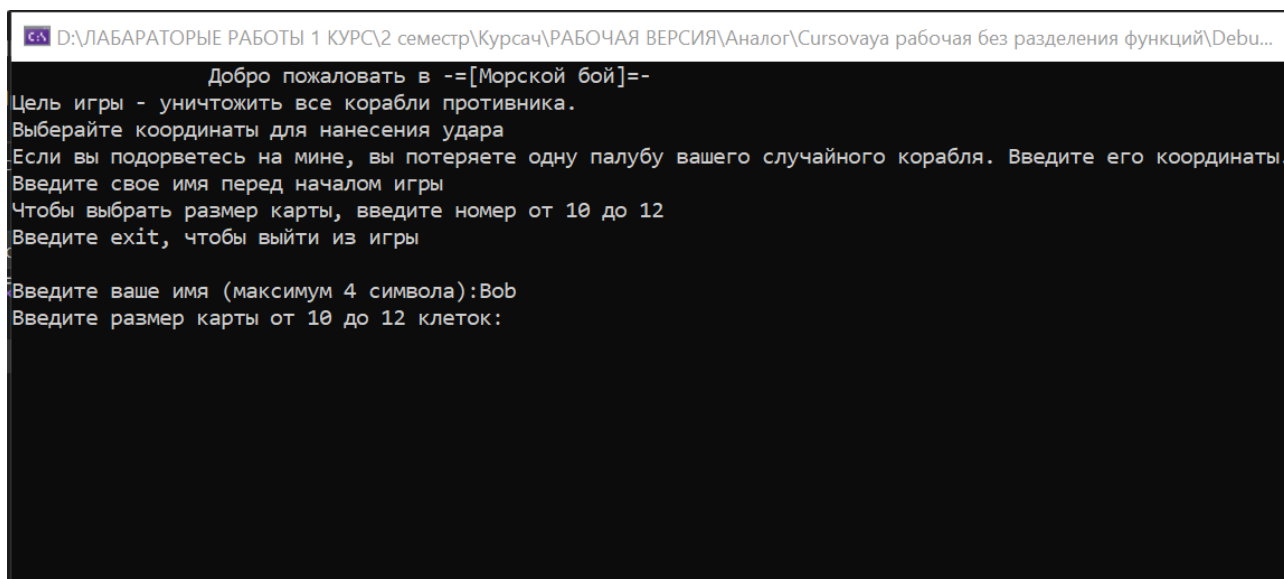


Рисунок 6 – Окно меню

После выбора уровня сложности открывается игровой уровень. В нем игрок. Вводя координаты, игрок наносит удары по полю противника. Удар может попасть по mine, тогда игроку придется указать координаты одной из своих палуб корабля, которая будет засчитана уничтоженной.



Рисунок 7 – Окно уровня игры

```

ПОЛЕ КОМПЬЮТЕРА
  А Б В Г Д Е Ё Ж З И К
1  О _ _ _ _ _ _ _ _ _
2  _ _ _ _ _ _ _ _ _
3  _ _ _ _ _ _ _ _ _
4  _ _ _ _ _ _ _ _ _
5  _ _ _ О О О О О _ _
6  _ _ _ О Х Х Х Х О _
7  _ О О О О О О О _ _
8  О О Х Х О _ _ _ _ _
9  _ О О О О _ _ _ _ _
10 _ _ _ _ _ _ _ _ _
11 _ _ _ _ _ _ _ _ _

ПОЛЕ ИГРОКА
  А Б В Г Д Е Ё Ж З И К
1  _ О О О _ _ _ _ _ _
2  М О Х О _ _ _ _ _ _
3  _ О Х О О _ _ _ _ О О
4  О О Х О _ _ _ _ О Х
5  О О О О О _ _ _ _ О О
6  О _ _ _ _ О О О _ _
7  _ _ _ _ _ О Х О _ _
8  _ _ _ _ _ О О О _ _
9  _ _ _ _ _ _ _ _ _
10 _ _ _ О _ _ О _ _ _
11 _ _ О _ _ _ _ _ _ _

введите координаты или 'стоп'

```

Рисунок 8 – Процесс игры

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Были освоены способы обработки событий от клавиатуры. Усвоены принципы реализации интерфейса. Получены базовые навыки программирования на языке С и Ассемблер. Изучены возможности среды программирования VisualStudio 2019. Получены навыки отладки и тестирования программ.

В рамках выполнения курсовой работы была написана программа (игра) морской бой. Программа не требует большого количества системных ресурсов, обеспечивает быструю обработку данных и мгновенный показ результата пользователю.

В дальнейшем программу можно улучшить, добавив более интерактивное меню, больше уровней сложности, отличающихся видом игрового поля, условиями проигрыша.

Список используемых источников

1. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.
2. К. Г. Финогенов. Основы языка Ассемблера. 1999 г.
3. Н. Г. Голубь. Искусство программирования на Ассемблере 2002 г.
4. К. Джемса. Учимся программировать на языке С++. 1997 г.

Приложение А Листинги программы

Приложение А.1 Файл «battleship.cpp»

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

int score = 2100;
int* Pscore = &score;
int win = 1;
int* Pwin = &win;
TPlayer players_score[10];

void leaderboard(TPlayer mas[10], int score, char name[5], FILE*
fps, char* name_s) {
    int score_fikt;
    char name_fikt[5];

    fps = fopen(name_s, "r");
    for (int i = 0; i < 10; i++) {
        fscanf(fps, "%s %d\n", &mas[i].name, &mas[i].score);
    }

    for (int k = 0; k < 10; k++) {
        if (mas[k].score < score) {
            score_fikt = mas[k].score;
            strncpy(name_fikt, mas[k].name, 5);
            mas[k].score = score;
            strncpy(mas[k].name, name, 5);
            score = score_fikt;
            strncpy(name, name_fikt, 5);
        }
    }
    fps = fopen(name_s, "w");
    for (int i = 0; i < 10; i++) {
        fprintf(fps, "%s %d\n", mas[i].name, mas[i].score);
    }
    fclose(fps);
}

int main() {
```

[illegible]

```

        system("cls");
        index10(Pscore, Pwin);
    }
    else if (result2 == 0)
    {
        system("cls");
        index11(Pscore, Pwin);
    }
    else if (result3 == 0)
    {
        system("cls");
        index12(Pscore, Pwin);
    }
    else {
        printf("\t\t\t\tInput error");
    }

    if (win == 1) {
        system("cls");
        leaderboard(players_score, score, name, fps, name_s);
        printf("Ваш счет %d\n", score);
    }

    Sleep(500);
    system("cls");
} while (ex == 0);
return 0;

return 0;
}

```

Приложение A.2

Файл «cur.h»

```

#pragma once

#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <windows.h>
#include <string.h>

typedef struct {
    char name[5];
    int score;
}TPlayer;

void pr(int* a, int size);
int index10(int *Pscore, int* Pwin);
int index11(int* Pscore, int* Pwin);
int index12(int* Pscore, int* Pwin);

```

Приложение А.3

Файл «asmpart.cpp»

```
#include "cur.h"

void pr(int* a, int size) {

    if (size == 10){
        printf("    А Б В Г Д Е Ё Ж З И\n");
    }
    else if(size == 11){
        printf("    А Б В Г Д Е Ё Ж З И К\n");
    }
    else if(size == 12){
        printf("    А Б В Г Д Е Ё Ж З И К Л\n");
    }

    for (int i = 0; i < size; i++) {
        printf(" %d", i + 1);
        if (i < 9) {
            printf(" ");
        }
        for (int j = 0; j < size; j++) {
            char p1[] = " _", p2[] = " O", p3[] = " M", p4[] = " X";
            int yy = *(a + i*size + j);
            __asm {
                mov eax, yy
                cmp eax, 0
                JNE skip111
                lea edx, p1;
                push edx;
                call printf
                pop edx
                jmp skip
                skip111 :

                cmp eax, 5
                JNE skip11
                lea edx, p1;
                push edx;
                call printf
                pop edx
                jmp skip
                skip11 :

                cmp eax, 8
                JNE skip1
                lea edx, p1;
                push edx;
                call printf
                pop edx
                jmp skip
                skip1 :

                cmp eax, 9
                JNE skip4
                lea edx, p1;
                push edx;
                call printf
                pop edx
                jmp skip
                skip4 :
            }
        }
    }
}
```

```

        mov eax, yy
        cmp eax, 3
        JNE skip2
        lea edx, p2;
    push edx;
    call printf
    pop edx
    jmp skip
    skip2 :

    mov eax, yy
    cmp eax, 4
    JNE skip3
    lea edx, p3;
    push edx;
    call printf
    pop edx
    jmp skip
    skip3 :

    mov eax, yy
    cmp eax, 6
    JNE skip
    lea edx, p4;
    push edx;
    call printf
    pop edx

    skip :
    }
}
printf("\n");
}

```

Приложение А.4

Файл «10.cpp»

```

#define _CRT_SECURE_NO_WARNINGS

#include "cur.h"

int a[10][10] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0} };
int z[10][10] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

```

```

        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } };
int b[20][3] = { {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
        {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
        {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
        {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0} };
int mz[10][2] = { {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0,
0}, {0, 0}, {0, 0} };
int c[10][3] = { {4, 0, 0}, {3, 0, 0}, {3, 0, 0}, {2, 0, 0}, {2, 0, 0},
        {2, 0, 0}, {1, 0, 0}, {1, 0, 0}, {1, 0, 0}, {1, 0, 0}
};
int m[3][3] = { {1, 0, 0}, {1, 0, 0}, {1, 0, 0} };

const int size = 10;
int num_of_hits1 = 0;
int* Phits = &num_of_hits1;

int* A = (int*)a, * Z = (int*)z, * B = (int*)b;

void SpawnOneShip1(int q, int* e) {
    int luck = 0;
    while (luck == 0) {
        int i = rand() % 2;
        if (i == 0) {
            int x = rand() % (size + 1 - q);
            int y = rand() % size;
            if (a[y][x + q - 1] == 0 and a[y][x] == 0) {
                if (x > 0) {
                    a[y][x - 1] = 5;
                    if (y > 0) {
                        a[y - 1][x - 1] = 5;
                    }
                    if (y < size-1) {
                        a[y + 1][x - 1] = 5;
                    }
                }
                if (x != size - q) {
                    a[y][x + q] = 5;
                    if (y > 0) {
                        a[y - 1][x + q] = 5;
                    }
                    if (y < size - 1) {
                        a[y + 1][x + q] = 5;
                    }
                }
            }
            for (int i = 0; i < q; i++) {
                a[y][i + x] = 8;
                b[*e][0] = i + x + 1;
                b[*e][1] = y + 1;
                *e = *e + 1;
                if (y > 0) {
                    a[y - 1][i + x] = 5;

```

```

        }
        if (y < size - 1) {
            a[y + 1][i + x] = 5;
        }
    }
    luck = 1;
}
else {
    int x = rand() % size;
    int y = rand() % (size + 1 - q);
    if (a[y + q - 1][x] == 0 and a[y][x] == 0) {
        if (y != 0 and y != 8) {
            a[y - 1][x] = 5;
            if (x > 0) {
                a[y - 1][x - 1] = 5;
            };
            if (x < size - 1) {
                a[y - 1][x + 1] = 5;
            };
        };
        if (y != 10 - q) {
            a[y + q][x] = 5;
            if (x > 0) {
                a[y + q][x - 1] = 5;
            };
            if (x < size - 17) {
                a[y + q][x + 1] = 5;
            };
        };
        for (int i = 0; i < q; i++) {
            a[i + y][x] = 8;
            b[*e][0] = x + 1;
            b[*e][1] = i + y + 1;
            *e = *e + 1;
            if (x > 0) {
                a[i + y][x - 1] = 5;
            };
            if (x < 9) {
                a[i + y][x + 1] = 5;
            };
        };
        luck = 1;
    };
};
};
}
void SpawnOneMinal(int q, int* e) { // q длина корабля извлекается из
    int luck = 0;
    while (luck == 0) {

        int x = rand() % (size + 1 - q);
        int y = rand() % size;
        if (a[y][x + q - 1] == 0 and a[y][x] == 0) {
            if (x > 0) {
                a[y][x - 1] = 5;
            };
        };
    };
}

```

```

        if (y > 0) {
            a[y - 1][x - 1] = 5;
        }
        if (y < size - 1) {
            a[y + 1][x - 1] = 5;
        }
    }
    if (x != size - q) {
        a[y][x + q] = 5;
        if (y > 0) {
            a[y - 1][x + q] = 5;
        }
        if (y < size - 1) {
            a[y + 1][x + q] = 5;
        }
    }
    for (int i = 0; i < q; i++) {
        a[y][i + x] = 9;

        if (y > 0) {
            a[y - 1][i + x] = 5;
        }
        if (y < size - 1) {
            a[y + 1][i + x] = 5;
        }
    }
    luck = 1;
}

};

};
void input1(int* PComp_ship_amount) {
    int ee = 0;
    int* e = &ee;
    for (int i = 0; i < 9; i++) {
        c[i + 1][2] = *PComp_ship_amount + c[i][0];
        *PComp_ship_amount = *PComp_ship_amount + c[i][0];
    }
    for (int i = 0; i < 10; i++) {
        SpawnOneShip1(c[i][0], e);
    }
    for (int i = 0; i < 3; i++) {
        SpawnOneMina1(m[i][0], e);
    }
}
void proverka1() {
    int r = 0;
    for (int i = 0; i < 10; i++) {
        int h = 0;
        for (int j = 0; j < c[i][0]; j++) {
            if (b[r][2] == 1)
                h += 1;
            r += 1;
        }
        if (c[i][0] == h and c[i][1] == 0) {
            for (int j = c[i][2]; j < c[i][0] + c[i][2]; j++) {

```

```

        a[b[j][1] - 1][b[j][0] - 1] = 6;
    }
    for (int y = 0; y < size; y++) {
        for (int x = 0; x < size; x++) {
            if (a[y][x] == 6) {
                if (x > 0 and a[y][x - 1] == 5)
                    a[y][x - 1] = 3;
                if (x < size - 1 and a[y][x + 1] == 5)
                    a[y][x + 1] = 3;
                if (y > 0 and a[y - 1][x] == 5)
                    a[y - 1][x] = 3;
                if (y < size - 1 and a[y + 1][x] == 5)
                    a[y + 1][x] = 3;
                if (x > 0 and y > 0 and a[y - 1][x - 1] == 5)
                    a[y - 1][x - 1] = 3;
                if (x > 0 and y < size - 1 and a[y + 1][x - 1] ==
5)
                    a[y + 1][x - 1] = 3;
                if (x < size - 1 and y > 0 and a[y - 1][x + 1] ==
5)
                    a[y - 1][x + 1] = 3;
                if (x < size - 1 and y < size - 1 and a[y + 1][x
+ 1] == 5)
                    a[y + 1][x + 1] = 3;
            }
        }
    }
    c[i][1] = 1;
}

}

}

void human1(int* PPodbil_hum_to_bot, char* Vvod_igroka, int* stop, int*
PPodbil_bot_to_hum) {
    int input_end = 0;
    printf("введите координаты или 'стоп'\n");
    while (input_end == 0 and *stop == 0) {
        int x = -1, y = -1;
        int correct_input = 0;

        while (correct_input == 0 and *stop == 0) {
            char XX[size];

            scanf("%s", &XX);
            fflush(stdin);
            if (strcmp(XX, "стоп") == 0) {
                *stop = 1;
            }
            else {
                if (strlen(XX) == 2) {
                    y = XX[1] - 49;
                }
                if (strlen(XX) == 3) {
                    y = (XX[1] - 48) * 10 + (XX[2] - 49);
                }
                if (y >= 0 and y <= size) {

```

```

        for (int i = 0; i < size; i++) {
            if (XX[0] == *(Vvod_igroka + 2 * i) or XX[0] ==
*(Vvod_igroka + 2 * i + 1)) {
                x = i;
                correct_input = 1;
                break;
            }
        }
    }
    if (correct_input == 0)
        printf("Внимание: недопустимые значения координат.\n");
}

if (*stop == 0) {
    if (a[y][x] == 6 or a[y][x] == 3 or a[y][x] == 4)
        printf("Внимание: вы уже попадали сюда.\n");

    if (a[y][x] == 8) {
        *Phits++;
        a[y][x] = 4;
        *PPodbil_hum_to_bot++;
        for (int i = 0; i < 20; i++) {
            if (b[i][1] == y + 1 and b[i][0] == x + 1)
                b[i][2] = 1;
        }

        proverkal();
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z, size);

        printf("введите координаты или 'стоп'\n");
    }
    if (a[y][x] == 0 or a[y][x] == 5) {
        *Phits++;
        input_end = 1;
        a[y][x] = 3;
        proverkal();
    }
    if (a[y][x] == 9) {
        correct_input = 0;
        a[y][x] = 3;
        *Phits++;
        printf("МИНА! введите координаты своего случайного
корабля\n");
        while (correct_input == 0 and *stop == 0) {
            char XX[10];

            scanf("%s", &XX);
            fflush(stdin);
            if (strcmp(XX, "стоп") == 0) {
                *stop = 1;
            }
        }
    }
}

```

```

    }
    else {
        if (strlen(XX) == 2) {
            y = XX[1] - 49;
        }
        if (strlen(XX) == 3) {
            y = (XX[1] - 48) * 10 + (XX[2] - 49);
        }
        if (y >= 0 and y <= size) {
            for (int i = 0; i < 10; i++) {
                if (XX[0] == *(Vvod_igroka + 2 * i) or
XX[0] == *(Vvod_igroka + 2 * i + 1)) {
                    x = i;
                    correct_input = 1;
                    break;
                }
            }
        }
        if (correct_input == 0)
            printf("Внимание: недопустимые значения
координат.\n");
    }
    z[y][x] = 4;
    *PPodbil_bot_to_hum++;

    proverkal();
    system("cls");
    printf("ПОЛЕ КОМПЬЮТЕРА\n");
    pr(A, size);
    printf("\nПОЛЕ ИГРОКА\n");
    pr(Z, size);

    printf("введите координаты или 'стоп'\n");

}

}

}

void mind1(int* ext_podb, int* ship_orientation, int* f, char* symbols,
int* stop, int* A3, int* PPodbil_hum_to_bot) {
    int input_end = 0;
    int x = 0, y = 0;
    while (input_end == 0) {
        while (1) {
            if (*ext_podb == 0) {
                x = rand() % size;
                y = rand() % size;
            }
            else {
                int tracking = 0;
                while (tracking == 0) {
                    int route = rand() % 4;

```

```

if (route <= 1 and *ship_orientation != 2) {
    mz[*ext_podb][0] = mz[0][0];
    x = mz[0][0];
    if (route == 0 and mz[*ext_podb - 1][1] > 0) {
        if (z[mz[*ext_podb - 1][1] - 1][x] == 0) {
            mz[*ext_podb][1] = mz[*ext_podb - 1][1]
- 1;

            y = mz[*ext_podb - 1][1] - 1;
            tracking = 1;
        }
        if (mz[0][1] < 9 and z[mz[*ext_podb - 1][1]
- 1][x] == 3) {

            mz[*ext_podb - 1][1] = mz[0][1];
            mz[*ext_podb - 1][0] = mz[0][0];
        }
        if (z[mz[*ext_podb - 1][1] - 1][x] == 4) {
            mz[*ext_podb - 1][1]--;
        }
    }
    if (route == 1 and mz[*ext_podb - 1][1] < 9) {
        if (z[mz[*ext_podb - 1][1] + 1][x] == 0) {
            mz[*ext_podb][1] = mz[*ext_podb - 1][1]
+ 1;

            y = mz[*ext_podb - 1][1] + 1;
            tracking = 1;
        }
        if (mz[0][1] > 0 and z[mz[*ext_podb - 1][1]
+ 1][x] == 3) {

            mz[*ext_podb - 1][1] = mz[0][1];
            mz[*ext_podb - 1][0] = mz[0][0];
        }
        if (z[mz[*ext_podb - 1][1] + 1][x] == 4) {
            mz[*ext_podb - 1][1]++;
        }
    }
}
if (route >= 2 and *ship_orientation != 1) {
    mz[*ext_podb][1] = mz[0][1];
    y = mz[*ext_podb][1];
    if (route == 2 and mz[*ext_podb - 1][0] > 0) {
        if (z[y][mz[*ext_podb - 1][0] - 1] == 0) {
            mz[*ext_podb][0] = mz[*ext_podb - 1][0]
- 1;

            x = mz[*ext_podb][0];
            tracking = 1;
        }
        if (mz[0][0] < 9 and z[y][mz[*ext_podb -
1][0] - 1] == 3) {

            mz[*ext_podb - 1][0] = mz[0][0];
            mz[*ext_podb - 1][1] = mz[0][1];
        }
        if (z[y][mz[*ext_podb - 1][0] - 1] == 4) {
            mz[*ext_podb - 1][0]--;
        }
    }
}
if (route == 3 and mz[*ext_podb - 1][0] < 9) {

```

```

        if (z[y][mz[*ext_podb - 1][0] + 1] == 0) {
            mz[*ext_podb][0] = mz[*ext_podb - 1][0]
+ 1;

            x = mz[*ext_podb][0];
            tracking = 1;
        }
        if (mz[0][0] > 0 and z[y][mz[*ext_podb -
1][0] + 1] == 3) {
            mz[*ext_podb - 1][0] = mz[0][0];
            mz[*ext_podb - 1][1] = mz[0][1];
        }
        if (z[y][mz[*ext_podb - 1][0] + 1] == 4) {
            mz[*ext_podb - 1][0]++;
        }
    }
}

    }
}
if (z[y][x] == 0)
    break;
}

printf("Captain PC: %c %d\n", *(symbols + x * 2), y + 1);
printf("Введите 'промах' 'попал' 'убил' 'мина'\n");

char otvet[20];
while (1) {

    scanf("%s", otvet);
    fflush(stdin);
    if (strcmp(otvet, "промах") == 0) {
        z[y][x] = 3;

        input_end = 1;
        break;
    }
    else {
        if (strcmp(otvet, "попал") == 0) {
            *f = *f + 1;
            z[y][x] = 4;
            mz[*ext_podb][0] = x;
            mz[*ext_podb][1] = y;
            *ext_podb = *ext_podb + 1;

            if (*ext_podb > 1 and *ship_orientation == 0) {
                if (abs(mz[0][1] - y) == 1)
                    *ship_orientation = 1;
                else
                    *ship_orientation = 2;
            }
            system("cls");
            printf("ПОЛЕ КОМПЬЮТЕРА\n");
            pr(A, size);
            printf("\nПОЛЕ ИГРОКА\n");
            pr(Z, size);
            break;
        }
    }
}

```

```

    }
    else {
        if (strcmp(otvet, "убил") == 0) {
            system("cls");
            pr(A, size);
            pr(Z, size);
            *f = *f + 1;
            z[y][x] = 6;
            for (int i = 0; i < 10; i++) {
                for (int y = 0; y < size; y++) {
                    for (int x = 0; x < size; x++) {
                        if (z[y][x] == 6) {
                            if (x > 0) {
                                if (z[y][x - 1] == 4)
                                    z[y][x - 1] = 6;
                                if (z[y][x - 1] == 0)
                                    z[y][x - 1] = 3;
                            }
                            if (x < size - 1) {
                                if (z[y][x + 1] == 4)
                                    z[y][x + 1] = 6;
                                if (z[y][x + 1] == 0)
                                    z[y][x + 1] = 3;
                            }
                            if (y > 0) {
                                if (z[y - 1][x] == 4)
                                    z[y - 1][x] = 6;
                                if (z[y - 1][x] == 0)
                                    z[y - 1][x] = 3;
                            }
                            if (y < size - 1) {
                                if (z[y + 1][x] == 4)
                                    z[y + 1][x] = 6;
                                if (z[y + 1][x] == 0)
                                    z[y + 1][x] = 3;
                            }
                            if (x > 0 and y > 0)
                                if (z[y - 1][x - 1] == 0)
                                    z[y - 1][x - 1] = 3;
                            if (x > 0 and y < size - 1)
                                if (z[y + 1][x - 1] == 0)
                                    z[y + 1][x - 1] = 3;
                            if (x < size - 1 and y > 0)
                                if (z[y - 1][x + 1] == 0)
                                    z[y - 1][x + 1] = 3;
                            if (x < size - 1 and y < size - 1)
                                if (z[y + 1][x + 1] == 0)
                                    z[y + 1][x + 1] = 3;
                        }
                    }
                }
            }
        }
    }
}

```



```

        while (*PPodbil_hum_to_bot <= *PComp_ship_amount and
*PPodbil_bot_to_hum <= *PComp_ship_amount and *PStop_game == 0) { //
проверка на окончание игры

        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z, size);
        human1(PPodbil_hum_to_bot, Vvod_igroka, PStop_game,
PPodbil_bot_to_hum);
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z, size);
        if (*PPodbil_hum_to_bot <= *PComp_ship_amount and
*PStop_game == 0) {
            mind1(ext_podb, ship_orientation, PPodbil_bot_to_hum,
Vvod_igroka, PStop_game, A, PPodbil_hum_to_bot);
        }
    }
    if (*PStop_game == 1) {

        printf("Выход в меню...");
    }
    else {
        system("cls");
        if (*PPodbil_bot_to_hum > *PPodbil_hum_to_bot) {
            printf("ПОРАЖЕНИЕ!!! Компьютер победил.\n");

            printf("\nПОЛЕ ИГРОКА\n");
            pr(Z, size);
        }
        else {
            printf("ПОБЕДА!!! Вы победили!!!\n");
            *Pscore = (((3*num_of_hits1 - *PPodbil_hum_to_bot) -
*PPodbil_bot_to_hum) * 10);
            *Pwin++;
            printf("ПОЛЕ КОМПЬЮТЕРА\n");
            pr(A, size);
        }
    }

    return 0;
}

```

Приложение A.5

Файл «11.cpp»

```
#define _CRT_SECURE_NO_WARNINGS
```

```

#include "cur.h"

int a2[11][11] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} };
int z2[11][11] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} };
int b2[20][3] = { {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
                  {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
                  {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
                  {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0} };
int mz2[10][2] = { {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0},
                   {0, 0}, {0, 0}, {0, 0}, {0, 0} };
int c2[10][3] = { {4, 0, 0}, {3, 0, 0}, {3, 0, 0}, {2, 0, 0}, {2, 0, 0},
                  {2, 0, 0}, {1, 0, 0}, {1, 0, 0}, {1, 0, 0}, {1, 0, 0} };
};
int m2[3][3] = { {1, 0, 0}, {1, 0, 0}, {1, 0, 0} };

const int size = 11;
int num_of_hits2 = 0;
int* Phits2 = &num_of_hits2;

int* A2 = (int*)a2, * Z2 = (int*)z2, * B2 = (int*)b2;

void SpawnOneShip2(int q, int* e) {
    int luck = 0;
    while (luck == 0) {
        int i = rand() % 2;
        if (i == 0) {
            int x = rand() % (size + 1 - q);
            int y = rand() % size;
            if (a2[y][x + q - 1] == 0 and a2[y][x] == 0) {
                if (x > 0) {

```

```

        a2[y][x - 1] = 5;
        if (y > 0) {
            a2[y - 1][x - 1] = 5;
        }
        if (y < size - 1) {
            a2[y + 1][x - 1] = 5;
        }
    }
    if (x != size - q) {
        a2[y][x + q] = 5;
        if (y > 0) {
            a2[y - 1][x + q] = 5;
        }
        if (y < size - 1) {
            a2[y + 1][x + q] = 5;
        }
    }
    for (int i = 0; i < q; i++) {
        a2[y][i + x] = 8;
        b2[*e][0] = i + x + 1;
        b2[*e][1] = y + 1;
        *e = *e + 1;
        if (y > 0) {
            a2[y - 1][i + x] = 5;
        }
        if (y < size - 1) {
            a2[y + 1][i + x] = 5;
        }
    }
    luck = 1;
}
}
else {
    int x = rand() % size;
    int y = rand() % (size + 1 - q);
    if (a2[y + q - 1][x] == 0 and a2[y][x] == 0) {
        if (y != 0 and y != 8) {
            a2[y - 1][x] = 5;
            if (x > 0) {
                a2[y - 1][x - 1] = 5;
            };
            if (x < size - 1) {
                a2[y - 1][x + 1] = 5;
            };
        };
        if (y != 10 - q) {
            a2[y + q][x] = 5;
            if (x > 0) {
                a2[y + q][x - 1] = 5;
            };
            if (x < size - 17) {
                a2[y + q][x + 1] = 5;
            };
        };
        for (int i = 0; i < q; i++) {
            a2[i + y][x] = 8;
            b2[*e][0] = x + 1;
            b2[*e][1] = i + y + 1;
            *e = *e + 1;
            if (x > 0) {
                a2[i + y][x - 1] = 5;
            };
            if (x < 9) {
                a2[i + y][x + 1] = 5;
            };
        };
    }
}

```

```

        };
        luck = 1;
    };
};

}
void SpawnOneMina2(int q, int* e) {
    int luck = 0;
    while (luck == 0) {

        int x = rand() % (size + 1 - q);
        int y = rand() % size;
        if (a2[y][x + q - 1] == 0 and a2[y][x] == 0) {
            if (x > 0) {
                a2[y][x - 1] = 5;
                if (y > 0) {
                    a2[y - 1][x - 1] = 5;
                }
                if (y < size - 1) {
                    a2[y + 1][x - 1] = 5;
                }
            }
            if (x != size - q) {
                a2[y][x + q] = 5;
                if (y > 0) {
                    a2[y - 1][x + q] = 5;
                }
                if (y < size - 1) {
                    a2[y + 1][x + q] = 5;
                }
            }
            for (int i = 0; i < q; i++) {
                a2[y][i + x] = 9;

                if (y > 0) {
                    a2[y - 1][i + x] = 5;
                }
                if (y < size - 1) {
                    a2[y + 1][i + x] = 5;
                }
            }
            luck = 1;
        }
    }

};

void input2(int* PComp_ship_amount) {
    int ee = 0;
    int* e = &ee;
    for (int i = 0; i < 9; i++) {
        c2[i + 1][2] = *PComp_ship_amount + c2[i][0];
        *PComp_ship_amount = *PComp_ship_amount + c2[i][0];
    }
    for (int i = 0; i < 10; i++) {
        SpawnOneShip2(c2[i][0], e);
    }
    for (int i = 0; i < 3; i++) {
        SpawnOneMina2(m2[i][0], e);
    }
}

void proverka2() {
    int r = 0;
    for (int i = 0; i < 10; i++) {
        int h = 0;
        for (int j = 0; j < c2[i][0]; j++) {

```

```

        if (b2[r][2] == 1)
            h += 1;
        r += 1;
    }
    if (c2[i][0] == h and c2[i][1] == 0) {
        for (int j = c2[i][2]; j < c2[i][0] + c2[i][2]; j++) {
            a2[b2[j][1] - 1][b2[j][0] - 1] = 6;
        }
        for (int y = 0; y < size; y++) {
            for (int x = 0; x < size; x++) {
                if (a2[y][x] == 6) {
                    if (x > 0 and a2[y][x - 1] == 5)
                        a2[y][x - 1] = 3;
                    if (x < size - 1 and a2[y][x + 1] == 5)
                        a2[y][x + 1] = 3;
                    if (y > 0 and a2[y - 1][x] == 5)
                        a2[y - 1][x] = 3;
                    if (y < size - 1 and a2[y + 1][x] == 5)
                        a2[y + 1][x] = 3;
                    if (x > 0 and y > 0 and a2[y - 1][x - 1] == 5)
                        a2[y - 1][x - 1] = 3;
                    if (x > 0 and y < size - 1 and a2[y + 1][x - 1] == 5)
                        a2[y + 1][x - 1] = 3;
                    if (x < size - 1 and y > 0 and a2[y - 1][x + 1] == 5)
                        a2[y - 1][x + 1] = 3;
                    if (x < size - 1 and y < size - 1 and a2[y + 1][x + 1] == 5)
                        a2[y + 1][x + 1] = 3;
                }
            }
        }
        c2[i][1] = 1;
    }
}

void human2(int* PPodbil_hum_to_bot, char* Vvod_igroka, int* stop, int*
PPodbil_bot_to_hum) {
    int input_end = 0;
    printf("Введите координаты или 'стоп'\n");
    while (input_end == 0 and *stop == 0) {
        int x = -1, y = -1;
        int correct_input = 0;

        while (correct_input == 0 and *stop == 0) {
            char XX[size];

            scanf("%s", &XX);
            fflush(stdin);
            if (strcmp(XX, "стоп") == 0) {
                *stop = 1;
            }
            else {
                if (strlen(XX) == 2) {
                    y = XX[1] - 49;
                }
                if (strlen(XX) == 3) {
                    y = (XX[1] - 48) * 10 + (XX[2] - 49);
                }
                if (y >= 0 and y <= size) {

```

```

        for (int i = 0; i < size; i++) {
            if (XX[0] == *(Vvod_igroka + 2 * i) or XX[0] ==
*(Vvod_igroka + 2 * i + 1)) {
                x = i;
                correct_input = 1;
                break;
            }
        }
    }
    if (correct_input == 0)
        printf("Внимание: недопустимые значения координат.\n");
}

if (*stop == 0) {
    if (a2[y][x] == 6 or a2[y][x] == 3 or a2[y][x] == 4)
        printf("Внимание: вы уже попадали сюда.\n");

    if (a2[y][x] == 8) {
        *Phits2++;
        a2[y][x] = 4;
        *PPodbil_hum_to_bot++;
        for (int i = 0; i < 20; i++) {
            if (b2[i][1] == y + 1 and b2[i][0] == x + 1)
                b2[i][2] = 1;
        }

        proverka2();
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A2, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z2, size);

        printf("введите координаты или 'стоп'\n");
    }
    if (a2[y][x] == 0 or a2[y][x] == 5) {
        *Phits2++;
        input_end = 1;
        a2[y][x] = 3;

        proverka2();
    }
    if (a2[y][x] == 9) {
        correct_input = 0;
        a2[y][x] = 3;
        *Phits2++;
        printf("МИНА! введите координаты своего случайного
корабля\n");
        while (correct_input == 0 and *stop == 0) {
            char XX[10];

```

```

scanf("%s", &XX);
fflush(stdin);
if (strcmp(XX, "стоп") == 0) {
    *stop = 1;
}
else {
    if (strlen(XX) == 2) {
        y = XX[1] - 49;
    }
    if (strlen(XX) == 3) {
        y = (XX[1] - 48) * 10 + (XX[2] - 49);
    }
    if (y >= 0 and y <= size) {
        for (int i = 0; i < 10; i++) {
            if (XX[0] == *(Vvod_igroka + 2 * i) or
XX[0] == *(Vvod_igroka + 2 * i + 1)) {
                x = i;
                correct_input = 1;
                break;
            }
        }
    }
    if (correct_input == 0)
        printf("Внимание: недопустимые значения
координат.\n");
}
z2[y][x] = 4;
*PPodbil_bot_to_hum++;

proverka2();
system("cls");
printf("ПОЛЕ КОМПЬЮТЕРА\n");
pr(A2, size);
printf("\nПОЛЕ ИГРОКА\n");
pr(Z2, size);

printf("введите координаты или 'стоп'\n");

}
}
}

void mind2(int* ext_podb, int* ship_orientation, int* f, char* symbols,
int* stop, int* A2, int* PPodbil_hum_to_bot) {
    int input_end = 0;
    int x = 0, y = 0;
    while (input_end == 0) {

```

```

while (1) {
    if (*ext_podb == 0) {
        x = rand() % size;
        y = rand() % size;
    }
    else {
        int tracking = 0;
        while (tracking == 0) {
            int route = rand() % 4;
            if (route <= 1 and *ship_orientation != 2) {
                mz2[*ext_podb][0] = mz2[0][0];
                x = mz2[0][0];
                if (route == 0 and mz2[*ext_podb - 1][1] > 0) {
                    if (z2[mz2[*ext_podb - 1][1] - 1][x] == 0) {
                        mz2[*ext_podb][1] = mz2[*ext_podb -
1][1] - 1;

                        y = mz2[*ext_podb - 1][1] - 1;
                        tracking = 1;
                    }
                    if (mz2[0][1] < 9 and z2[mz2[*ext_podb -
1][1] - 1][x] == 3) {

                        mz2[*ext_podb - 1][1] = mz2[0][1];
                        mz2[*ext_podb - 1][0] = mz2[0][0];
                    }
                    if (z2[mz2[*ext_podb - 1][1] - 1][x] == 4) {
                        mz2[*ext_podb - 1][1]--;
                    }
                }
                if (route == 1 and mz2[*ext_podb - 1][1] < 9) {
                    if (z2[mz2[*ext_podb - 1][1] + 1][x] == 0) {
                        mz2[*ext_podb][1] = mz2[*ext_podb -
1][1] + 1;

                        y = mz2[*ext_podb - 1][1] + 1;
                        tracking = 1;
                    }
                    if (mz2[0][1] > 0 and z2[mz2[*ext_podb -
1][1] + 1][x] == 3) {

                        mz2[*ext_podb - 1][1] = mz2[0][1];
                        mz2[*ext_podb - 1][0] = mz2[0][0];
                    }
                    if (z2[mz2[*ext_podb - 1][1] + 1][x] == 4) {
                        mz2[*ext_podb - 1][1]++;
                    }
                }
            }
            if (route >= 2 and *ship_orientation != 1) {
                mz2[*ext_podb][1] = mz2[0][1];
                y = mz2[*ext_podb][1];
                if (route == 2 and mz2[*ext_podb - 1][0] > 0) {
                    if (z2[y][mz2[*ext_podb - 1][0] - 1] == 0) {
                        mz2[*ext_podb][0] = mz2[*ext_podb -
1][0] - 1;

```

```

        x = mz2[*ext_podb][0];
        tracking = 1;
    }
    if (mz2[0][0] < 9 and z2[y][mz2[*ext_podb -
1][0] - 1] == 3) {
        mz2[*ext_podb - 1][0] = mz2[0][0];
        mz2[*ext_podb - 1][1] = mz2[0][1];
    }
    if (z2[y][mz2[*ext_podb - 1][0] - 1] == 4) {
        mz2[*ext_podb - 1][0]--;
    }
}
if (route == 3 and mz2[*ext_podb - 1][0] < 9) {
    if (z2[y][mz2[*ext_podb - 1][0] + 1] == 0) {
        mz2[*ext_podb][0] = mz2[*ext_podb -
1][0] + 1;
        x = mz2[*ext_podb][0];
        tracking = 1;
    }
    if (mz2[0][0] > 0 and z2[y][mz2[*ext_podb -
1][0] + 1] == 3) {
        mz2[*ext_podb - 1][0] = mz2[0][0];
        mz2[*ext_podb - 1][1] = mz2[0][1];
    }
    if (z2[y][mz2[*ext_podb - 1][0] + 1] == 4) {
        mz2[*ext_podb - 1][0]++;
    }
}
}
}
if (z2[y][x] == 0)
    break;
}

printf("Captain PC: %c %d\n", *(symbols + x * 2), y + 1);
printf("Введите 'промах' 'попал' 'убил' 'мина'\n");

char otvet[20];
while (1) {
    scanf("%s", otvet);
    fflush(stdin);
    if (strcmp(otvet, "промах") == 0) {
        z2[y][x] = 3;

        input_end = 1;
        break;
    }
    else {
        if (strcmp(otvet, "попал") == 0) {
            *f = *f + 1;

```

```

z2[y][x] = 4;
mz2[*ext_podb][0] = x;
mz2[*ext_podb][1] = y;
*ext_podb = *ext_podb + 1;

if (*ext_podb > 1 and *ship_orientation == 0) {
    if (abs(mz2[0][1] - y) == 1)
        *ship_orientation = 1;
    else
        *ship_orientation = 2;
}
system("cls");
printf("ПОЛЕ КОМПЬЮТЕРА\n");
pr(A2, size);
printf("\nПОЛЕ ИГРОКА\n");
pr(Z2, size);
break;
}
else {
    if (strcmp(otvet, "убил") == 0) {
        system("cls");
        pr(A2, size);
        pr(Z2, size);
        *f = *f + 1;
        z2[y][x] = 6;
        for (int i = 0; i < 10; i++) {
            for (int y = 0; y < size; y++) {
                for (int x = 0; x < size; x++) {
                    if (z2[y][x] == 6) {
                        if (x > 0) {
                            if (z2[y][x - 1] == 4)
                                z2[y][x - 1] = 6;
                            if (z2[y][x - 1] == 0)
                                z2[y][x - 1] = 3;
                        }
                        if (x < size) {
                            if (z2[y][x + 1] == 4)
                                z2[y][x + 1] = 6;
                            if (z2[y][x + 1] == 0)
                                z2[y][x + 1] = 3;
                        }
                    }
                    if (y > 0) {
                        if (z2[y - 1][x] == 4)
                            z2[y - 1][x] = 6;
                        if (z2[y - 1][x] == 0)
                            z2[y - 1][x] = 3;
                    }
                    if (y < size) {
                        if (z2[y + 1][x] == 4)
                            z2[y + 1][x] = 6;
                        if (z2[y + 1][x] == 0)
                            z2[y + 1][x] = 3;
                    }
                }
            }
        }
    }
}

```

```

        }
        if (x > 0 and y > 0)
            if (z2[y - 1][x - 1] == 0)

z2[y - 1][x - 1] = 3;

        if (x > 0 and y < size)
            if (z2[y + 1][x - 1] == 0)

z2[y + 1][x - 1] = 3;

        if (x < size and y > 0)
            if (z2[y - 1][x + 1] == 0)

z2[y - 1][x + 1] = 3;

        if (x < size and y < size)
            if (z2[y + 1][x + 1] == 0)

z2[y + 1][x + 1] = 3;
    }
}

}
}
*ship_orientation = 0;
*ext_podb = 0;
break;
}
else {
    if (strcmp(otvet, "мина") == 0) {
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A2, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z2, size);

        int boom = 0;
        while (boom == 0) {
            x = rand() % size;
            y = rand() % size;

            if (a2[y][x] == 8) {
                a2[y][x] = 4;
                *PPodbil_hum_to_bot++;

                boom = boom++;
            }
        }

        *ship_orientation = 0;
        *ext_podb = 0;
        break;
    }
    printf("Доступные варианты ответа: промах,
попал, убил, мина.\n");
}

```

```

        }
    }
}

}

int index11(int* Pscore, int* Pwin) {

    srand(time(NULL));
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    int* PPodbil_bot_to_hum = new int(0);
    int* PStop_game = new int(0);
    int* PComp_ship_amount = new int(0);
    int* PPodbil_hum_to_bot = new int(0);
    int* ship_orientation = new int(0);
    int* ext_podb = new int(0);
    char vvod_igroka[] = "АаБбВвГгДдЕеЁёЖжЗзИиКкЛл";

    char* Vvod_igroka = vvod_igroka;
    input2(PComp_ship_amount);

    while (*PPodbil_hum_to_bot <= *PComp_ship_amount and
*PPodbil_bot_to_hum <= *PComp_ship_amount and *PStop_game == 0) {

        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A2, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z2, size);
        human2(PPodbil_hum_to_bot, Vvod_igroka, PStop_game,
PPodbil_bot_to_hum);
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A2, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z2, size);
        if (*PPodbil_hum_to_bot <= *PComp_ship_amount and *PStop_game ==
0) {
            mind2(ext_podb, ship_orientation, PPodbil_bot_to_hum,
Vvod_igroka, PStop_game, A2, PPodbil_hum_to_bot);
        }
    }
    if (*PStop_game == 1) {

        printf("Выход в меню...");
    }
    else {

```

```

system("cls");
if (*PPodbil_bot_to_hum > *PPodbil_hum_to_bot) {
    printf("ПОРАЖЕНИЕ!!! Компьютер победил.\n");

    printf("\nПОЛЕ ИГРОКА\n");
    pr(Z2, size);
}
else {
    printf("ПОВЕДА!!! Вы победили!!!\n");
    *Pscore = (((3 * num_of_hits2 - *PPodbil_hum_to_bot) -
*PPodbil_bot_to_hum)*10);
    *Pwin++;
    printf("ПОЛЕ КОМПЬЮТЕРА\n");
    pr(A2, size);

}
}

return 0;
}

```

Приложение А.6

Файл «12.cpp»

```

#define _CRT_SECURE_NO_WARNINGS

#include "cur.h"

int a3[12][12] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} };
int z3[12][12] = { {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} };
int b3[20][3] = { {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
    {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
    {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0},
    {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0} };

```

```

int mz3[10][2] = { {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0} };
int c3[10][3] = { {4, 0, 0}, {3, 0, 0}, {3, 0, 0}, {2, 0, 0}, {2, 0, 0},
                  {2, 0, 0}, {1, 0, 0}, {1, 0, 0}, {1, 0, 0}, {1, 0, 0} };
int m3[3][3] = { {1, 0, 0}, {1, 0, 0}, {1, 0, 0} };

const int size = 12;
int num_of_hits3 = 0;
int* Phits3 = &num_of_hits3;

int* A3 = (int*)a3, * Z3 = (int*)z3, * B3 = (int*)b3;

void SpawnOneShip3(int q, int* e) {
    int luck = 0;
    while (luck == 0) {
        int i = rand() % 2;
        if (i == 0) {
            int x = rand() % (size + 1 - q);
            int y = rand() % size;
            if (a3[y][x + q - 1] == 0 and a3[y][x] == 0) {
                if (x > 0) {
                    a3[y][x - 1] = 5;
                    if (y > 0) {
                        a3[y - 1][x - 1] = 5;
                    }
                    if (y < 9) {
                        a3[y + 1][x - 1] = 5;
                    }
                }
                if (x != 10 - q) {
                    a3[y][x + q] = 5;
                    if (y > 0) {
                        a3[y - 1][x + q] = 5;
                    }
                    if (y < 9) {
                        a3[y + 1][x + q] = 5;
                    }
                }
            }
            for (int i = 0; i < q; i++) {
                a3[y][i + x] = 8;
                b3[*e][0] = i + x + 1;
                b3[*e][1] = y + 1;
                *e = *e + 1;
                if (y > 0) {
                    a3[y - 1][i + x] = 5;
                }
                if (y < 9) {
                    a3[y + 1][i + x] = 5;
                }
            }
            luck = 1;
        }
        else {
            int x = rand() % size;
            int y = rand() % (size + 1 - q);
            if (a3[y + q - 1][x] == 0 and a3[y][x] == 0) {
                if (y != 0 and y != 8) {
                    a3[y - 1][x] = 5;
                    if (x > 0) {
                        a3[y - 1][x - 1] = 5;
                    }
                }
            }
        }
    }
}

```

```

        if (x < 9) {
            a3[y - 1][x + 1] = 5;
        };
    };
    if (y != 10 - q) {
        a3[y + q][x] = 5;
        if (x > 0) {
            a3[y + q][x - 1] = 5;
        };
        if (x < 9) {
            a3[y + q][x + 1] = 5;
        };
    };
    for (int i = 0; i < q; i++) {
        a3[i + y][x] = 8;
        b3[*e][0] = x + 1;
        b3[*e][1] = i + y + 1;
        *e = *e + 1;
        if (x > 0) {
            a3[i + y][x - 1] = 5;
        };
        if (x < 9) {
            a3[i + y][x + 1] = 5;
        };
    };
    luck = 1;
};
};
};
}

void SpawnOneMina3(int q, int* e) {
    int luck = 0;
    while (luck == 0) {

        int x = rand() % (size + 1 - q);
        int y = rand() % size;
        if (a3[y][x + q - 1] == 0 and a3[y][x] == 0) {
            if (x > 0) {
                a3[y][x - 1] = 5;
                if (y > 0) {
                    a3[y - 1][x - 1] = 5;
                }
                if (y < 9) {
                    a3[y + 1][x - 1] = 5;
                }
            }
            if (x != size - q) {
                a3[y][x + q] = 5;
                if (y > 0) {
                    a3[y - 1][x + q] = 5;
                }
                if (y < 9) {
                    a3[y + 1][x + q] = 5;
                }
            }
        }
        for (int i = 0; i < q; i++) {
            a3[y][i + x] = 9;

            if (y > 0) {
                a3[y - 1][i + x] = 5;
            }
            if (y < 9) {
                a3[y + 1][i + x] = 5;
            }
        }
    }
}

```

```

        }
    }
    luck = 1;
}

};

};
void input3(int* PComp_ship_amount) {
    int ee = 0;
    int* e = &ee;
    for (int i = 0; i < 9; i++) {
        c3[i + 1][2] = *PComp_ship_amount + c3[i][0];
        *PComp_ship_amount = *PComp_ship_amount + c3[i][0];
    }
    for (int i = 0; i < 10; i++) {
        SpawnOneShip3(c3[i][0], e);
    }
    for (int i = 0; i < 3; i++) {
        SpawnOneMina3(m3[i][0], e);
    }
}

void proverka3() {
    int r = 0;
    for (int i = 0; i < 10; i++) {
        int h = 0;
        for (int j = 0; j < c3[i][0]; j++) {
            if (b3[r][2] == 1)
                h += 1;
            r += 1;
        }
        if (c3[i][0] == h and c3[i][1] == 0) {
            for (int j = c3[i][2]; j < c3[i][0] + c3[i][2]; j++) {
                a3[b3[j][1] - 1][b3[j][0] - 1] = 6;
            }
            for (int y = 0; y < size; y++) {
                for (int x = 0; x < size; x++) {
                    if (a3[y][x] == 6) {
                        if (x > 0 and a3[y][x - 1] == 5)
                            a3[y][x - 1] = 3;
                        if (x < 9 and a3[y][x + 1] == 5)
                            a3[y][x + 1] = 3;
                        if (y > 0 and a3[y - 1][x] == 5)
                            a3[y - 1][x] = 3;
                        if (y < 9 and a3[y + 1][x] == 5)
                            a3[y + 1][x] = 3;
                        if (x > 0 and y > 0 and a3[y - 1][x - 1] == 5)
                            a3[y - 1][x - 1] = 3;
                        if (x > 0 and y < 9 and a3[y + 1][x - 1] == 5)
                            a3[y + 1][x - 1] = 3;
                        if (x < 9 and y > 0 and a3[y - 1][x + 1] == 5)
                            a3[y - 1][x + 1] = 3;
                        if (x < 9 and y < 9 and a3[y + 1][x + 1] == 5)
                            a3[y + 1][x + 1] = 3;
                    }
                }
            }
            c3[i][1] = 1;
        }
    }
}

void human3(int* PPodbil_hum_to_bot, char* Vvod_igroka, int* stop, int*
PPodbil_bot_to_hum) {
    int input_end = 0;

```

```

printf("введите координаты или 'стоп'\n");
while (input_end == 0 and *stop == 0) {
    int x = -1, y = -1;
    int correct_input = 0;

    while (correct_input == 0 and *stop == 0) {
        char XX[size];

        scanf("%s", &XX);
        fflush(stdin);
        if (strcmp(XX, "стоп") == 0) {
            *stop = 1;
        }
        else {
            if (strlen(XX) == 2) {
                y = XX[1] - 49;
            }
            if (strlen(XX) == 3) {
                y = (XX[1] - 48) * 10 + (XX[2] - 49);
            }
            if (y >= 0 and y <= size) {
                for (int i = 0; i < size; i++) {
                    if (XX[0] == *(Vvod_igroka + 2 * i) or XX[0] == *(Vvod_igroka + 2
* i + 1)) {
                        x = i;
                        correct_input = 1;
                        break;
                    }
                }
            }
        }
        if (correct_input == 0)
            printf("Внимание: недопустимые значения координат.\n");
    }

    if (*stop == 0) {
        if (a3[y][x] == 6 or a3[y][x] == 3 or a3[y][x] == 4)
            printf("Внимание: вы уже попадали сюда.\n");

        if (a3[y][x] == 8) {
            *Phits3++;
            a3[y][x] = 4;
            *PPodbil_hum_to_bot++;
            for (int i = 0; i < 20; i++) {
                if (b3[i][1] == y + 1 and b3[i][0] == x + 1)
                    b3[i][2] = 1;
            }

            proverka3();
            system("cls");
            printf("ПОЛЕ КОМПЬЮТЕРА\n");
            pr(A3, size);
            printf("\nПОЛЕ ИГРОКА\n");
            pr(Z3, size);

            printf("введите координаты или 'стоп'\n");
        }
        if (a3[y][x] == 0 or a3[y][x] == 5) {
            *Phits3++;
            input_end = 1;
            a3[y][x] = 3;
        }
    }
}

```

```

        proverka3();
    }
    if (a3[y][x] == 9) {
        correct_input = 0;
        a3[y][x] = 3;
        *Phits3++;
        printf("МИНА! введите координаты своего случайного корабля\n");
        while (correct_input == 0 and *stop == 0) {
            char XX[10];

            scanf("%s", &XX);
            fflush(stdin);
            if (strcmp(XX, "стоп") == 0) {
                *stop = 1;
            }
            else {
                if (strlen(XX) == 2) {
                    y = XX[1] - 49;
                }
                if (strlen(XX) == 3) {
                    y = (XX[1] - 48) * 10 + (XX[2] - 49);
                }
                if (y >= 0 and y <= size) {
                    for (int i = 0; i < 10; i++) {
                        if (XX[0] == *(Vvod_igroka + 2 * i) or XX[0] ==
*(Vvod_igroka + 2 * i + 1)) {
                            x = i;
                            correct_input = 1;
                            break;
                        }
                    }
                }
            }
            if (correct_input == 0)
                printf("Внимание: недопустимые значения координат.\n");
        }
        z3[y][x] = 4;
        *PPodbil_bot_to_hum++;

        proverka3();
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A3, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z3, size);

        printf("введите координаты или 'стоп'\n");
    }
}

}

}

void mind3(int* ext_podb, int* ship_orientation, int* f, char* symbols, int* stop, int*
A3, int* PPodbil_hum_to_bot) {
    int input_end = 0;
    int x = 0, y = 0;
    while (input_end == 0) {
        while (1) {
            if (*ext_podb == 0) {
                x = rand() % size;
                y = rand() % size;
            }
        }
    }
}

```

```

    }
    else {
        int tracking = 0;
        while (tracking == 0) {
            int route = rand() % 4;
            if (route <= 1 and *ship_orientation != 2) {
                mz3[*ext_podb][0] = mz3[0][0];
                x = mz3[0][0];
                if (route == 0 and mz3[*ext_podb - 1][1] > 0) {
                    if (z3[mz3[*ext_podb - 1][1] - 1][x] == 0) {
                        mz3[*ext_podb][1] = mz3[*ext_podb - 1][1] - 1;
                        y = mz3[*ext_podb - 1][1] - 1;
                        tracking = 1;
                    }
                    if (mz3[0][1] < 9 and z3[mz3[*ext_podb - 1][1] - 1][x] == 3)
                    {
                        mz3[*ext_podb - 1][1] = mz3[0][1];
                        mz3[*ext_podb - 1][0] = mz3[0][0];
                    }
                    if (z3[mz3[*ext_podb - 1][1] - 1][x] == 4) {
                        mz3[*ext_podb - 1][1]--;
                    }
                }
                if (route == 1 and mz3[*ext_podb - 1][1] < 9) {
                    if (z3[mz3[*ext_podb - 1][1] + 1][x] == 0) {
                        mz3[*ext_podb][1] = mz3[*ext_podb - 1][1] + 1;
                        y = mz3[*ext_podb - 1][1] + 1;
                        tracking = 1;
                    }
                    if (mz3[0][1] > 0 and z3[mz3[*ext_podb - 1][1] + 1][x] == 3)
                    {
                        mz3[*ext_podb - 1][1] = mz3[0][1];
                        mz3[*ext_podb - 1][0] = mz3[0][0];
                    }
                    if (z3[mz3[*ext_podb - 1][1] + 1][x] == 4) {
                        mz3[*ext_podb - 1][1]++;
                    }
                }
            }
            if (route >= 2 and *ship_orientation != 1) {
                mz3[*ext_podb][1] = mz3[0][1];
                y = mz3[*ext_podb][1];
                if (route == 2 and mz3[*ext_podb - 1][0] > 0) {
                    if (z3[y][mz3[*ext_podb - 1][0] - 1] == 0) {
                        mz3[*ext_podb][0] = mz3[*ext_podb - 1][0] - 1;
                        x = mz3[*ext_podb][0];
                        tracking = 1;
                    }
                    if (mz3[0][0] < 9 and z3[y][mz3[*ext_podb - 1][0] - 1] == 3)
                    {
                        mz3[*ext_podb - 1][0] = mz3[0][0];
                        mz3[*ext_podb - 1][1] = mz3[0][1];
                    }
                    if (z3[y][mz3[*ext_podb - 1][0] - 1] == 4) {
                        mz3[*ext_podb - 1][0]--;
                    }
                }
                if (route == 3 and mz3[*ext_podb - 1][0] < 9) {
                    if (z3[y][mz3[*ext_podb - 1][0] + 1] == 0) {
                        mz3[*ext_podb][0] = mz3[*ext_podb - 1][0] + 1;
                        x = mz3[*ext_podb][0];
                        tracking = 1;
                    }
                }
            }
        }
    }
}

```

```

        if (mz3[0][0] > 0 and z3[y][mz3[*ext_podb - 1][0] + 1] == 3)
    {
        mz3[*ext_podb - 1][0] = mz3[0][0];
        mz3[*ext_podb - 1][1] = mz3[0][1];
    }
    if (z3[y][mz3[*ext_podb - 1][0] + 1] == 4) {
        mz3[*ext_podb - 1][0]++;
    }
    }
    }
    }
    if (z3[y][x] == 0)
        break;
}

printf("Captain PC: %c %d\n", *(symbols + x * 2), y + 1);
printf("Введите 'промах' 'попал' 'убил' 'мина'\n");

char otvet[20];
while (1) {

    scanf("%s", otvet);
    fflush(stdin);
    if (strcmp(otvet, "промах") == 0) {
        z3[y][x] = 3;

        input_end = 1;
        break;
    }
    else {
        if (strcmp(otvet, "попал") == 0) {
            *f = *f + 1;
            z3[y][x] = 4;
            mz3[*ext_podb][0] = x;
            mz3[*ext_podb][1] = y;
            *ext_podb = *ext_podb + 1;

            if (*ext_podb > 1 and *ship_orientation == 0) {
                if (abs(mz3[0][1] - y) == 1)
                    *ship_orientation = 1;
                else
                    *ship_orientation = 2;
            }
            system("cls");
            printf("ПОЛЕ КОМПЬЮТЕРА\n");
            pr(A3, size);
            printf("\nПОЛЕ ИГРОКА\n");
            pr(Z3, size);
            break;
        }
        else {
            if (strcmp(otvet, "убил") == 0) {
                system("cls");
                pr(A3, size);
                pr(Z3, size);
                *f = *f + 1;
                z3[y][x] = 6;
                for (int i = 0; i < 10; i++) {
                    for (int y = 0; y < size; y++) {
                        for (int x = 0; x < size; x++) {
                            if (z3[y][x] == 6) {
                                if (x > 0) {
                                    if (z3[y][x - 1] == 4) z3[y][x - 1] = 6;

```

```

        if (z3[y][x - 1] == 0) z3[y][x - 1] = 3;
    }
    if (x < size) {
        if (z3[y][x + 1] == 4) z3[y][x + 1] = 6;
        if (z3[y][x + 1] == 0) z3[y][x + 1] = 3;
    }
    if (y > 0) {
        if (z3[y - 1][x] == 4) z3[y - 1][x] = 6;
        if (z3[y - 1][x] == 0) z3[y - 1][x] = 3;
    }
    if (y < size) {
        if (z3[y + 1][x] == 4) z3[y + 1][x] = 6;
        if (z3[y + 1][x] == 0) z3[y + 1][x] = 3;
    }
    if (x > 0 and y > 0)
        if (z3[y - 1][x - 1] == 0) z3[y - 1][x - 1]
= 3;

    if (x > 0 and y < size)
        if (z3[y + 1][x - 1] == 0) z3[y + 1][x - 1]
= 3;

    if (x < size and y > 0)
        if (z3[y - 1][x + 1] == 0) z3[y - 1][x + 1]
= 3;

    if (x < size and y < size)
        if (z3[y + 1][x + 1] == 0) z3[y + 1][x + 1]
= 3;

    }
    }
}
*ship_orientation = 0;
*ext_podb = 0;
break;
}
else {
    if (strcmp(otvet, "мина") == 0) {
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A3, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z3, size);

        int boom = 0;
        while (boom == 0) {
            x = rand() % size;
            y = rand() % size;

            if (a3[y][x] == 8) {
                a3[y][x] = 4;
                *PPodbil_hum_to_bot++;

                boom = boom++;
            }
        }

        *ship_orientation = 0;
        *ext_podb = 0;
        break;
    }
    printf("Доступные варианты ответа: промах, попал, убил,
мина.\nц");

```

```

    }
    }
}

int index12(int *Pscore, int* Pwin) {

    srand(time(NULL));
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    int* PPodbil_bot_to_hum = new int(0);
    int* PStop_game = new int(0);
    int* PComp_ship_amount = new int(0);
    int* PPodbil_hum_to_bot = new int(0);
    int* ship_orientation = new int(0);
    int* ext_podb = new int(0);
    char vvod_igroka[] = "АаВбВвГгДдЕеЁёЖжЗзИиКкЛл";

    char* Vvod_igroka = vvod_igroka;

    input3(PComp_ship_amount);

    while (*PPodbil_hum_to_bot <= *PComp_ship_amount and *PPodbil_bot_to_hum <=
    *PComp_ship_amount and *PStop_game == 0) {

        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A3, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z3, size);
        human3(PPodbil_hum_to_bot, Vvod_igroka, PStop_game, PPodbil_bot_to_hum);
        system("cls");
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A3, size);
        printf("\nПОЛЕ ИГРОКА\n");
        pr(Z3, size);
        if (*PPodbil_hum_to_bot <= *PComp_ship_amount and *PStop_game == 0) {
            mind3(ext_podb, ship_orientation, PPodbil_bot_to_hum, Vvod_igroka,
PStop_game, A3, PPodbil_hum_to_bot);
        }
    }
    if (*PStop_game == 1) {

        printf("Выход в меню...");
    }
    else {
        system("cls");
        if (*PPodbil_bot_to_hum > *PPodbil_hum_to_bot) {
            printf("ПОРАЖЕНИЕ!!! Компьютер победил.\n");

            printf("\nПОЛЕ ИГРОКА\n");
            pr(Z3, size);
        }
        else {
            printf("ПОБЕДА!!! Вы победили!!!\n");
        }
    }
}

```

```

        *Pscore = (((3 * num_of_hits3 - *PPodbil_hum_to_bot) - *PPodbil_bot_to_hum) *
10);
        *Pwin++;
        printf("ПОЛЕ КОМПЬЮТЕРА\n");
        pr(A3, size);
    }
}

return 0;
}

```

Приложение В

Снимки экрана



Рисунок 9 - Заставка

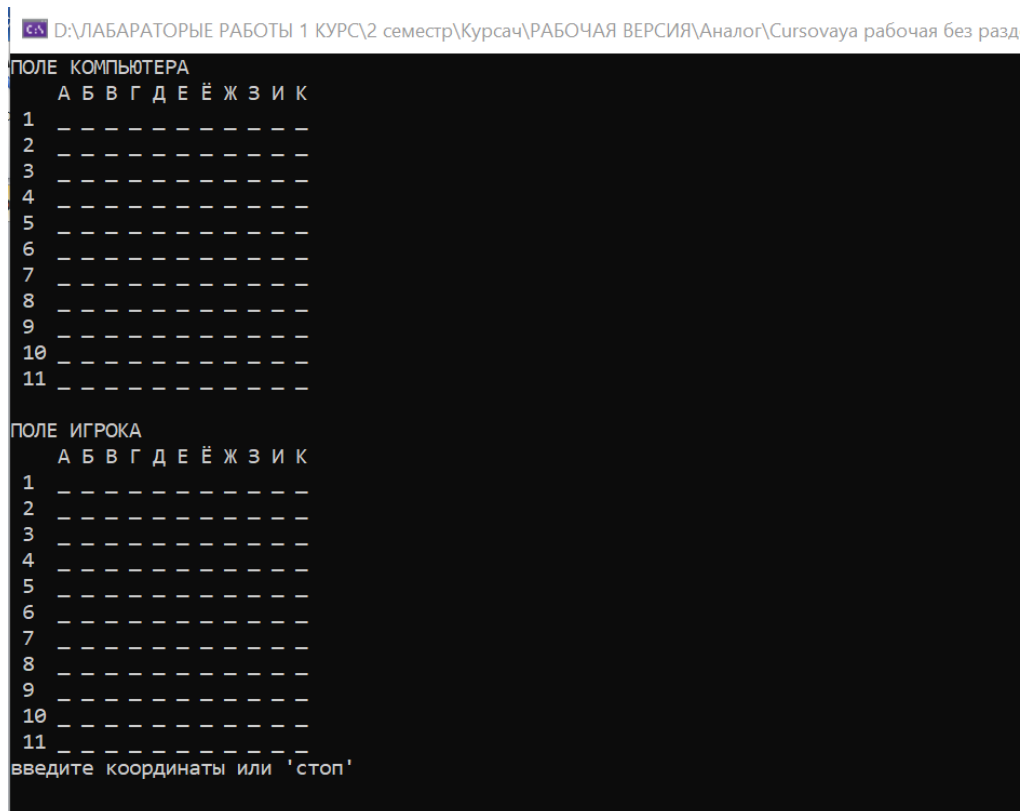


Рисунок 10 – Игровое поле

```

ПОЛЕ КОМПЬЮТЕРА
  А Б В Г Д Е Ё Ж З И К
1  О _ _ _ _ _ _ _ _ _
2  _ _ _ _ _ _ _ _ _
3  _ _ _ _ _ _ _ _ _
4  _ _ _ _ _ _ _ _ _
5  _ _ _ О О О О О О _
6  _ _ _ О Х Х Х Х О _
7  _ О О О О О О О О _
8  О О Х Х О _ _ _ _
9  _ О О О О _ _ _ _
10 _ _ _ _ _ _ _ _ _
11 _ _ _ _ _ _ _ _ _

ПОЛЕ ИГРОКА
  А Б В Г Д Е Ё Ж З И К
1  _ О О О _ _ _ _ _
2  М О Х О _ _ _ _ _
3  _ О Х О О _ _ _ О О
4  О О Х О _ _ _ _ О Х
5  О О О О О _ _ _ О О
6  О _ _ _ _ О О О _ _
7  _ _ _ _ _ О Х О _ _
8  _ _ _ _ _ О О О _ _
9  _ _ _ _ _ _ _ _ _
10 _ _ _ О _ _ О _ _
11 _ _ О _ _ _ _ _ _

введите координаты или 'стоп'

```

Рисунок 11 – Игровое поле во время игры

Приложение С

Результат работы программы

Приложение С.1

Файл s.txt

ads 1240
asw 1190
dsa 1120
was 1070
rew 1050
usb 1030
ter 970
pop 950
pit 920
rul 910
obo 890