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

Пояснительная записка

к курсовой работе
по курсу «Программирование»
на тему «Разработка программы сложной структуры методом нисходящего программирования»

Выполнил:

Студент группы
21ВВ3
Савицкий Макар

Приняли:

Голотенко Н. О.

Пенза 2022

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 Ловицкану Макара Алексеевича Группа 21ВВ3

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

Исходные данные (технические требования) на проектирование

Приложение "игра морской бой". Приложение должно автоматически расставлять корабли на поле, определять подбитые корабли и определять победителя. Программа должна быть написана на языке си и включать ассемблер. Режим работы видеосистемы - текстовый. Устройства ввода - клавиатура, мышь.

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 "25" марта 2022

Дата защиты проекта "09" 06. 22

Руководитель Патушина Д.В., Головченко М.О.

Задание получил "25" марта 2022 г.

Студент Савицкий Макар Алексеевич

Содержание

Введение 5
1. Постановка задачи и обязательные требования 6
2. Выбор решения.....7
3. Описание разработки программы 7
4. Отладка и тестирование.....8
5. Описание программы.....9
5.1 Функции.....9
5.2 Разработка функций на языке Assembler.....10
6. Руководство пользователя 11
Заключение.....15
Список используемых источников.....16
Приложение A. Иерархическая структура программы.....17
Приложение B. Схема данных программы.....18
Приложение C. Листинг программы.....19
Приложение D. Схемы функций программы 28
Приложение E. Схема взаимодействия программы.....32

Введение

Компьютерные технологии уже давно укоренились в нашей жизни. Многие привычные для нас вещи были бы невозможны 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 вкупе с ассемблерными вставками. Разработка велась в среде Microsoft Visual Studio 2019.

1. Постановка задачи и обязательные требования

Необходимо разработать программу «Игра змейка», которая должна отвечать следующим требованиям:

1. Многомодульность. Программа должна быть поделена на логические модули. Это упростит дальнейшее совершенствование программы, а также позволит локализовать ошибки.

2. Использование сложных типов данных: массивов, структур, объектов и т.д. Для разработки данной игры необходимо подробно изучить принципы работы с массивами, структурами и объектами 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, с заставкой.

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

Приложение морской бой состоит из двух модулей:

1. proger.cpp

2. pr.cpp

Файл «proger.cpp» является основным модулем данной курсовой работы, которая контролирует основные функции и вызов других функций из других файлов. Он связан заголовочным файлом «stdafx.h».

Файл «pr.cpp» используется для вывода поля на консоль.

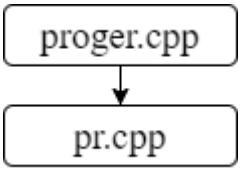

Рисунок 1 — Структура модулей программы

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.

Интерфейс игры будет представлен в виде 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. Прогресс игры может быть сохранен в документе «s.txt» во время хода игрока.

Основная программа proger.cpp выполняется функцией main.

4. 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 остановки, 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.

5. Описание программы

5.1 Функции

Приложение proger.cpp является основным модулем программы. После запуска программы выводится вопрос продолжить ли прошлую игру. После программа ответа, затем отрисовывается игровое поле, и игра начинается.

qqq – отвечает за расстановку одного кораблика заданного размера в случайном допустимом месте на карте.

proverka – отвечает за проверку убитых кораблей на поле бота, находит по координатам убитый корабль и обводит его разведанной зоной.

print – служебная команда, отвечает за вывод поля в исходном виде, не скрывая корабли. Не задействована в самой игре.

human – отвечает взаимодействие игрока с полем бота путем ввода координат, так же отвечает за сохранение игры.

mind – отвечает за логику всей игры. Так как игрок не вводит свое поле в базу данных, боту самостоятельно приходится выслеживать корабли. В начале координаты выбираются случайным образом. При попадании начинается отслеживание кораблика, попадая по соседним клеткам. После второго попадания стрельба ведется в уже установленной горизонтали. После убийства кораблика вокруг него устанавливается зона, где точно не могут быть другие корабли.
loading – отвечает за загрузку сохранённой игры, загружая поле бота и его прошлые результаты разведки поля игрока.

end – отвечает за окончание игры и выводит поле проигравшего.

save – отвечает за сохранение основной информации в файл для дальнейшего продолжения игры.

main – отвечает за основную игру, собирая все функции в одно целое.

В файле «pr.cpp» находится функция pr , которая отображает поля на экран.

5.2 Разработка функций на языке Assembler

Как известно, функции, реализованные на языке ассемблера, занимают меньше места и работают быстрее. В качестве разрабатываемых функций были выбраны функции для обнуления поля, хода игрока и проверки.

Описание функций выполнено на языке С, а вычисления (частично, в некоторых случаях приходилось использовать язык СИ) представляют собой ассемблерную вставку. Для реализации необходимо перед такой вставкой прописать ключевое слово «__asm», а сам код расположить в фигурных скобках.

Были использованы следующие ассемблерные инструкции:

- команда mov используется для копирования значения из источника в приемник;

- команда cmp производит сравнение двух чисел;

- команда lea создает указатель на элемент;

- команды push и pop для обращения к стеку;

- условный переход jne при ZF != 0;

- безусловный переход jmp;

```
__asm {
  mov eax, yy
  cmp eax, 0
  JNE skip111
  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 :

  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 :
  }
```

В зависимости от введенных данных выводим тот или иной символ на экран.

6. Руководство пользователя

Программа proger предназначена для игры в морской бой. Она имеет понятный интерфейс и интуитивно понятное управление.

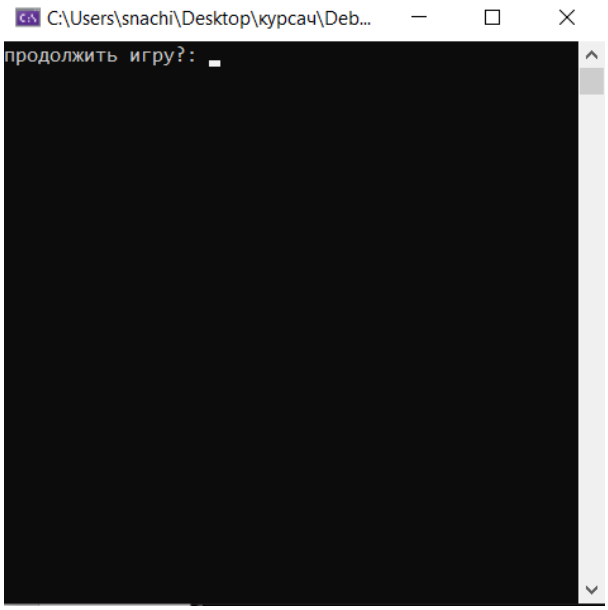

Рисунок 2.1 — Новая игра?

Вводится решение игрока и нажимается Enter.

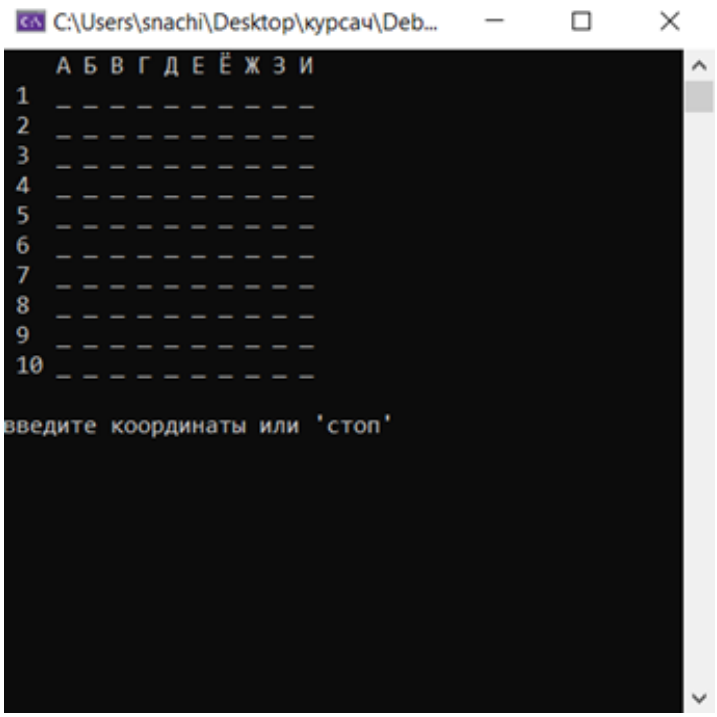

Рисунок 3.2 — Ввод поля бота

Отображается поле игры, случайным образом расположенные корабли скрыты.

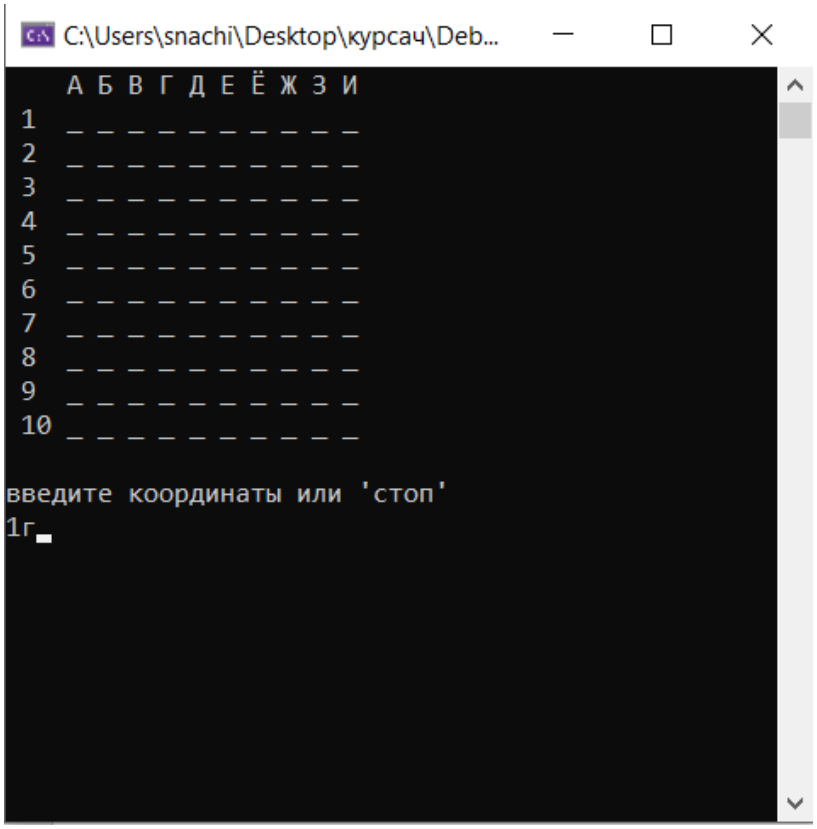

Рисунок 3.3 — ввод

Вводятся координаты от игрока.

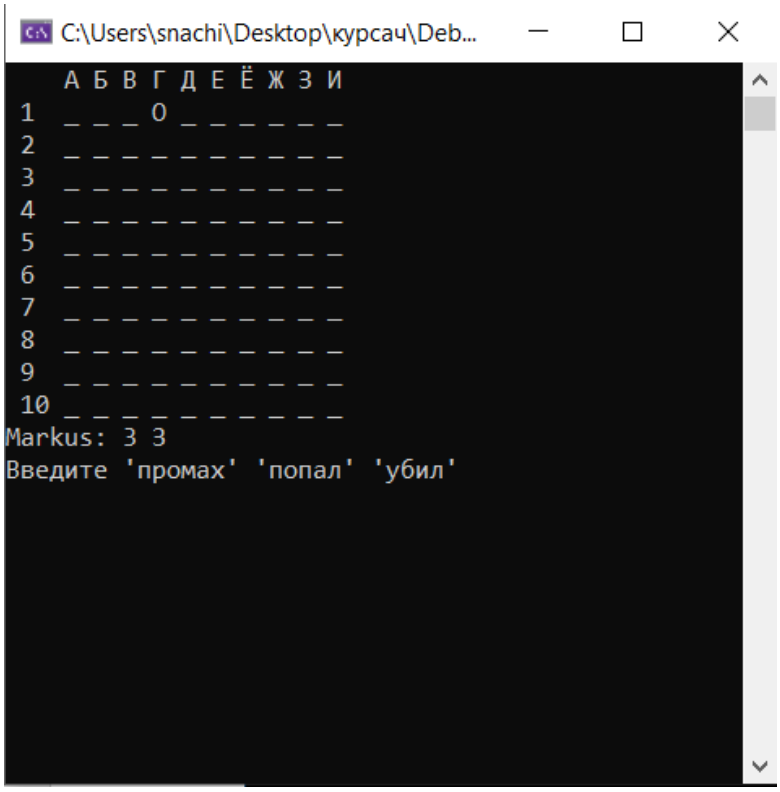

Рисунок 3.4 — Ход игры

Результаты выстрела сразу выводятся на экран, а так же бот называет вам координаты и так до конца игры

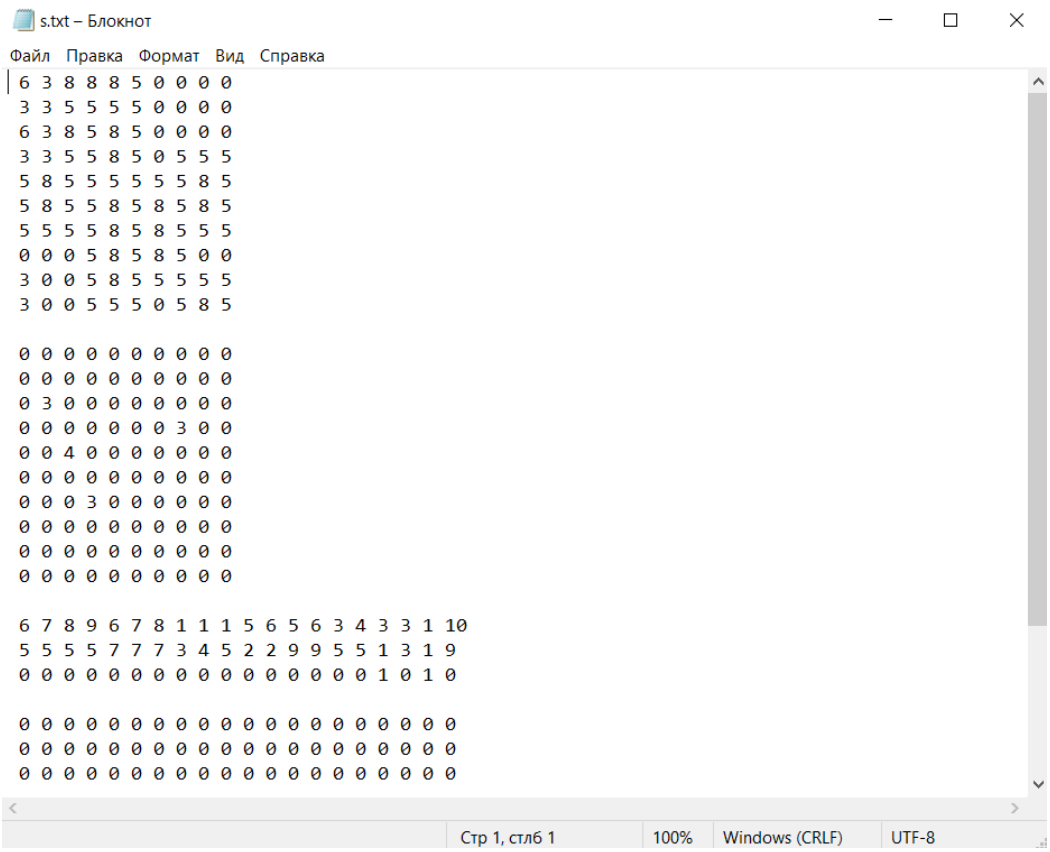

Рисунок 3.5 — Просмотр сохраненной

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 возможность 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.

Программу можно улучшить, оптимизировав код для более быстрого выполнения функций и разнообразив игру.

Список используемых источников

1. https://learnc.info/c/functions.html

2. http://natalia.appmat.ru/c&c++/assembler.html

3. http://www.codenet.ru/progr/cpp/spr/141.php

4. https://prog-cpp.ru/c-pointers/

Приложение А. Иерархическая структура программы

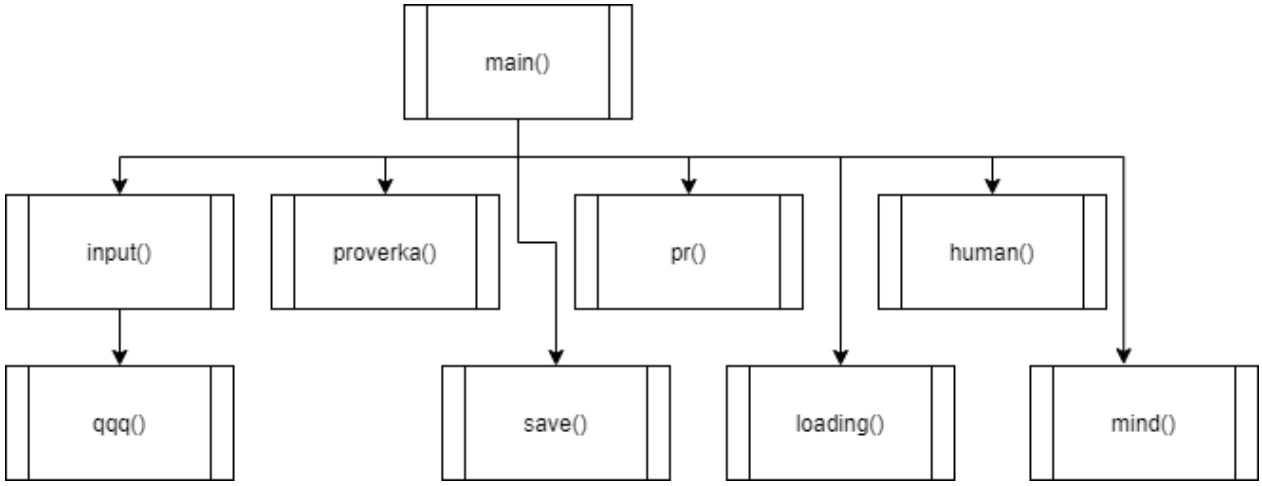

Рисунок 4 — Иерархическая структура программы

Приложение В. Схема данных программы

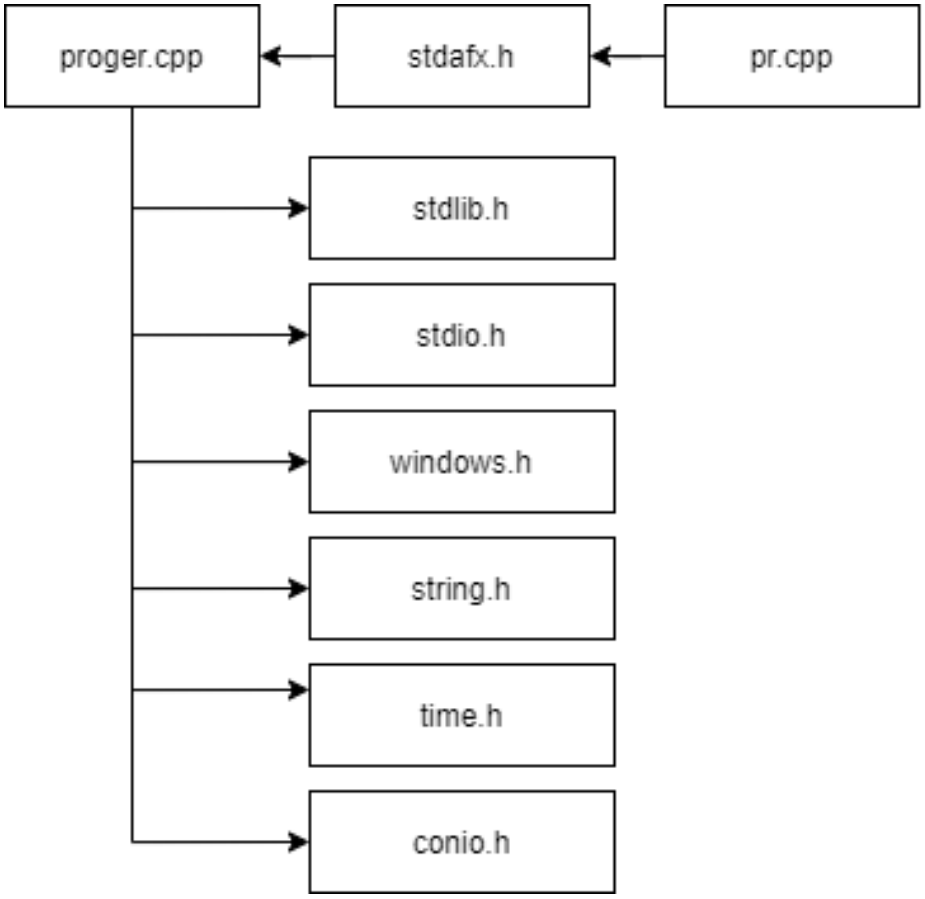

Рисунок 5 — Схема данных программы

Приложение С. Листинг программы

stdafx.h

```
#pragma once

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

void pr(int* a);
```

proger.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include "stdafx.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 };
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,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
int ms[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, 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* A = (int*)a, * Z = (int*)z, * B = (int*)b, * MS = (int*)ms, * MZ = (int*)mz, * C = (int*)c;

void qqq(int q, int* e) {
   int luck = 0;
   while (luck == 0) {
      int i = rand() % 2;
      if (i == 0) {
         int x = rand() % (11 - q);
         int y = rand() % 10;
         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 < 9) {
                  a[y + 1][x - 1] = 5;
               }
            }
            if (x != 10 - q) {
               a[y][x + q] = 5;
               if (y > 0) {
                  a[y - 1][x + q] = 5;
               }
               if (y < 9) {
                  a[y + 1][x + q] = 5;
               }
            }
            for (int i = 0; i < q; i++) {
               a[y][i + x] = 8;
               b[*e][0] = y + 1;
```

```
            b[*e][1] = i + x + 1;
            *e = *e + 1;
            if (y > 0) {
               a[y - 1][i + x] = 5;
            }
            if (y < 9) {
               a[y + 1][i + x] = 5;
            }
         }
         luck = 1;
      }
   }
   else {
      int x = rand() % 10;
      int y = rand() % (11 - 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 < 9) {
               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 < 9) {
               a[y + q][x + 1] = 5;
            };
         };
         for (int i = 0; i < q; i++) {
            a[i + y][x] = 8;
            b[*e][0] = i + y + 1;
            b[*e][1] = x + 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 input(int* endd) {
    int ee = 0;
    int* e = &ee;
    for (int i = 0; i < 9; i++) {
        c[i + 1][2] = *endd + c[i][0];
        *endd = *endd + c[i][0];
    }
    for (int i = 0; i < 10; i++) {
        qqq(c[i][0], e);
    }
}
void proverka() {
    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][0] - 1][b[j][1] - 1] = 6;
            }
            for (int y = 0; y < 10; y++) {
                for (int x = 0; x < 10; x++) {
                    if (a[y][x] == 6) {
```

```
                if (x > 0 and a[y][x - 1] == 5)
                    a[y][x - 1] = 3;
                if (x < 9 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 < 9 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 < 9 and a[y + 1][x - 1] == 5)
                    a[y + 1][x - 1] = 3;
                if (x < 9 and y>0 and a[y - 1][x + 1] == 5)
                    a[y - 1][x + 1] = 3;
                if (x < 9 and y < 9 and a[y + 1][x + 1] == 5)
                    a[y + 1][x + 1] = 3;
            }
        }
    }
    c[i][1] = 1;
  }
 }
}
void print(int* a) {
  for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
      printf(" %d", *(a + i * 10 + j));
    };
    printf("\n");
  };
}
void human(int* w, char* qweqwe, char* qweqwe1, int* stop) {
  int ddds = 0;
  while (ddds == 0){
    int x = 0, y = 0;
    int av = 0;
    while (av == 0) {
      char XX[10] = "11", X;
      printf("введите координаты или 'стоп'\n");
```

```
    scanf("%s", &XX);
    fflush(stdin);
    if (strcmp(XX, "стоп") == 0) {
        *stop = 1;
        av = 1;
        ddds = 1;
    }
    else {
        if (strlen(XX) == 2 or strlen(XX) == 3) {
            if (strlen(XX) == 2){
                y = XX[0] - 49;
                X = XX[1];
            }
            else {
                y = (XX[0] - 48)*10 + (XX[1] - 49);
                X = XX[2];
            }
            if (y >= 0 and y <= 9) {
                for (int i = 0; i < 10; i++) {
                    if (X == *(qweqwe + 2 * i) or X == *(qweqwe1 + i)) {
                        x = qweqwe[2 * i + 1] - 48;
                        av =
                            1;
                        break;
                    }
                }
            }
        }
        if (a[y][x] == 3 or a[y][x] == 6) {
            ddds = 1;
        }
    }
    if (av != 1) {
        printf("Внимание: недопустимые значения координат.\n");
    }

}
if (*stop == 0){
    if (a[y][x] == 8) {
```

```
            a[y][x] = 4;
            *w = *w + 1;
            ddds = 1;
            for (int i = 0; i < 20; i++) {
               if (b[i][0] == y + 1 and b[i][1] == x + 1)
                  b[i][2] = 1;
            }
         }
         else {
            if (a[y][x] == 0 or a[y][x] == 5) {
               ddds = 1;
               a[y][x] = 3;
            }
         }
      }
   }
}
void mind(int* n, int* f, char* qweqwe,int * stop) {
   int ddds = 0;
   while (ddds == 0) {
      int opa;
      int x = 0, y = 0, t = 0, k = 0, pp = 0, s = 0, luck = 0, sd = 0, m = 0;
      while (luck == 0) {
         int d = rand() % 100 + 1;
         if (m > 0 and d <= 6 + 80 * 3) {
            int v = 0;
            while (v == 0) {
               int o = rand() % 4 + 1;
               if (o < 3 and *n != 1) {
                  mz[m][0] = mz[0][0];
                  x = mz[m][0];
                  if (o == 1 and mz[m - 1][1] > 0) {
                     if (z[mz[m - 1][1] - 1][x] == 0 and sd == 0) {
                        mz[m][1] = mz[m - 1][1] - 1;
                        y = mz[m - 1][1] - 1;
                        v = 1;
                     }
                     if ((mz[0][1] < 9 and sd == 1) or mz[m - 1][1] == 9 or z[mz[m - 1][1] - 1][x] == 3) {
                        mz[m - 1][1] = mz[0][1];
```

```
            mz[m - 1][0] = mz[0][0];
            sd = 0;
        }
        if (z[mz[m - 1][1] - 1][x] == 4) {
            mz[m - 1][1] = mz[m - 1][1] - 1;
        }
    }
    if (o == 2 and mz[m - 1][1] < 9) {
        if (z[mz[m - 1][1] + 1][x] == 0 and sd == 0) {
            mz[m][1] = mz[m - 1][1] + 1;
            y = mz[m - 1][1] + 1;
            v = 1;
        }
        if ((mz[0][1] > 0 and sd == 1) or mz[m - 1][1] == 0 or z[mz[m - 1][1] + 1][x] == 3) {
            mz[m - 1][1] = mz[0][1];
            mz[m - 1][0] = mz[0][0];
            sd = 0;
        }
        if (z[mz[m - 1][1] + 1][x] == 4) {
            mz[m - 1][1] = mz[m - 1][1] + 1;
        }
    }
}
if (o > 2 and *n != 2) {
    mz[m][1] = mz[0][1];
    y = mz[m][1];
    if (o == 3 and mz[m - 1][0] > 0) {
        if (z[y][mz[m - 1][0] - 1] == 0 and sd == 0) {
            mz[m][0] = mz[m - 1][0] - 1;
            x = mz[m][0];
            v = 1;
        }
        if ((mz[0][0] < 9 and sd == 1) or mz[m - 1][0] == 9 or z[y][mz[m - 1][0] - 1] == 3) {
            mz[m - 1][0] = mz[0][0];
            mz[m - 1][1] = mz[0][1];
            sd = 0;
        }
        if (z[y][mz[m - 1][0] - 1] == 4) {
            mz[m - 1][0] = mz[m - 1][0] - 1;
```

```
            }
          }
          if (o == 4 and mz[m - 1][0] < 9) {
            if (z[y][mz[m - 1][0] + 1] == 0 and sd == 0) {
              mz[m][0] = mz[m - 1][0] + 1;
              x = mz[m][0];
              v = 1;
            }
            if ((mz[0][0] > 0 and sd == 1) or mz[m - 1][0] == 0 or z[y][mz[m - 1][0] + 1] == 3) {
              mz[m - 1][0] = mz[0][0];
              mz[m - 1][1] = mz[0][1];
              sd = 0;
            }
            if (z[y][mz[m - 1][0] + 1] == 4) {
              mz[m - 1][0] = mz[m - 1][0] + 1;
            }
          }
        }
      }
    }
    else {
      x = rand() % 10;
      y = rand() % 10;
      if (m > 0) {
        s = 1;
        k = k + 1;
        int ere = 0;
        while (ere == 0) {
          int ghg = 0;
          x = rand() % 10;
          y = rand() % 10;
          if (y > 0 and z[y - 1][x] == 4) {
            ghg = ghg + 1;
          }
          if (y < 9 and z[y + 1][x] == 4) {
            ghg = ghg + 1;
          }
          if (x > 0 and z[y][x - 1] == 4) {
            ghg = ghg + 1;
```

```
            }
            if (x < 9 and z[y][x + 1] == 4) {
               ghg = ghg + 1;
            }
            if (y > 0 and x > 0 and z[y - 1][x - 1] == 4) {
               ghg = ghg + 1;
            }
            if (y < 9 and x>0 and z[y + 1][x - 1] == 4) {
               ghg = ghg + 1;
            }
            if (y > 0 and x < 9 and z[y - 1][x + 1] == 4) {
               ghg = ghg + 1;
            }
            if (y < 9 and x < 9 and z[y + 1][x + 1] == 4) {
               ghg = ghg + 1;
            }
            if (ghg == 0) {
               ere = 1;
            }
         }
      }
   }
   if (z[y][x] == 0) {
      luck = 1;
      printf("Markus: %d %c\n", y + 1, *(qweqwe + x * 2));
      t = 0;
      char g[20];
      while (t == 0) {
         printf("Введите 'промах' 'попал' 'убил'\n");
         scanf("%s", g);
         fflush(stdin);
         if (strcmp(g, "промах") == 0) {
            t = 1;
            z[y][x] = 3;
            if (s == 0) {
               if (m > 1 and *n != 0)
                  sd = 1;
               ddds = 1;
            }
```

```
}
else {
   if (strcmp(g, "попал") == 0) {
      ddds = 1;
      *f = *f + 1;
      t = 1;
      z[y][x] = 4;
      if (s == 0) {
         mz[m][0] = x;
         mz[m][1] = y;
         m = m + 1;
         if (m > 1 and *n == 0) {
            if (abs(mz[0][1] - y) == 1)
               *n = 2;
            else
               *n = 1;
         }
      }
      else {
         int pt = 0;
         while (pt == 0) {
            opa = rand() % 20;
            if (ms[opa][2] == 0) {
               ms[opa][0] = x;
               ms[opa][1] = y;
               ms[opa][2] = 1;
               pt = 1;
            }
         }
         pp = pp + 1;
      }
   }
   else {
      if (strcmp(g, "убил") == 0) {
         ddds = 1;
         *f = *f + 1;
         t = 1;
         z[y][x] = 6;
         for (int i = 0; i < 10; i++) {
```

```
for (int y = 0; y < 10; y++) {
   for (int x = 0; x < 10; 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 < 9) {
            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 < 9) {
            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] == 4)
               z[y - 1][x - 1] = 6;
            if (z[y - 1][x - 1] == 0)
               z[y - 1][x - 1] = 3;
         }
         if (x > 0 and y < 9) {
            if (z[y + 1][x - 1] == 4)
               z[y + 1][x - 1] = 6;
            if (z[y + 1][x - 1] == 0)
               z[y + 1][x - 1] = 3;
         }
```

```
            if (x < 9 and y>0) {
               if (z[y - 1][x + 1] == 4)
                  z[y - 1][x + 1] = 6;
               if (z[y - 1][x + 1] == 0)
                  z[y - 1][x + 1] = 3;
            }
            if (x < 9 and y < 9) {
               if (z[y + 1][x + 1] == 4)
                  z[y + 1][x + 1] = 6;
               if (z[y + 1][x + 1] == 0)
                  z[y + 1][x + 1] = 3;
            }
         }
      }
   }
}
if (s == 0) {
   n = 0;
   m = 0;
   sd = 0;
   k = 0;
   if (pp > 0) {
      m = 1;
      pp = pp - 1;
      int pt = 0;
      while (pt == 0) {
         opa = rand() % 20;
         if (ms[opa][2] == 1) {
            if (z[ms[opa][1]][ms[opa][0]] == 4) {
               mz[0][0] = ms[opa][0];
               mz[0][1] = ms[opa][1];
               ms[opa][2] = 0;
               ms[opa][1] = 0;
               ms[opa][0] = 0;
               pt = 1;
            }
            else {
               ms[opa][2] = 0;
               ms[opa][1] = 0;
```

```
                                ms[opa][0] = 0;
                            }
                        }
                    }
                }
            }
            else {
                printf("Доступные варианты ответа: промах, попал, убил.\nц");
            }
        }
    }
}
}
void end(int* f, int* w) {
    int questio = 0;
    if (*f > *w) {
        printf("Markus: добро пожаловать в Detroit\n");
        pr(Z);
    }
    else {
        printf("Markus: что ж это твоя победа\n");
        pr(A);
    }
}
void save(FILE* fps, char* name_s,int *w, int *f) {
    fps = fopen(name_s, "w");
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            fprintf(fps, " %d", *(A + 10 * i + j));

        }
        fprintf(fps, "\n");
    }
    fprintf(fps, "\n");
    for (int i = 0; i < 10; i++) {
```

```
            for (int j = 0; j < 10; j++) {
                fprintf(fps, " %d", *(Z + 10 * i + j));
            }
            fprintf(fps, "\n");
        }
        fprintf(fps, "\n");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 20; j++) {
                fprintf(fps, " %d", *(B + 3 * j + i));
            }
            fprintf(fps, "\n");
        }
        fprintf(fps, "\n");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 20; j++) {
                fprintf(fps, " %d", *(MS + 3 * j + i));
            }
            fprintf(fps, "\n");
        }
        fprintf(fps, "\n");
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 10; j++) {
                fprintf(fps, " %d", *(MZ + 2 * j + i));
            }
            fprintf(fps, "\n");
        }
        fprintf(fps, "\n");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 10; j++) {
                fprintf(fps, " %d", *(C + 3 * j + i));
            }
            fprintf(fps, "\n");
        }
        fprintf(fps, " %d", *w);
        fprintf(fps, " %d", *f);
    }
    void loading(FILE* fps, char* name_s, int* w, int* f) {
        int t = 0;
        char g[20];
```

```
while (t == 0) {
    printf("продолжить игру?: ");
    scanf("%s", &g);
    fflush(stdin);
    if (strcmp(g, "да") == 0) {
        fps = fopen(name_s, "r");
        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                fscanf(fps, "%d", &*(A + 10 * i + j));
            }
        }
        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                fscanf(fps, "%d", &*(Z + 10 * i + j));
            }
        }
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 20; j++) {
                fscanf(fps, "%d", &*(B + 3 * j + i));
            }
        }
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 20; j++) {
                fscanf(fps, "%d", &*(MS + 3 * j + i));
            }
        }
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 10; j++) {
                fscanf(fps, "%d", &*(MZ + 2 * j + i));
            }
        }
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 10; j++) {
                fscanf(fps, "%d", &*(C + 3 * j + i));
            }
        }
        fscanf(fps, " %d", &*w);
        fscanf(fps, " %d", &*f);
        t = 1;
```

```
        }
        if (strcmp(g, "нет") == 0) {
            t = 1;
        }
    }
}
void main() {

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

    FILE* fps;
    int ddds = 0, nn = 0, ww = 0, ff = 0, enddd = 0, stopp = 0;
    int * n = &nn, * w = &ww, * f = &ff, * endd = &enddd, * stop = &stopp;
    char qweqwee[] = "А0Б1В2Г3Д4Е5Ё6Ж7З8И9";
    char qweqwe11[] = "абвгдеёжзи";
    char name_ss[] = "C:/Users/snachi/Desktop/курсач/s.txt";
    char* qweqwe = qweqwee, * qweqwe1 = qweqwe11, * name_s = name_ss;

    if ((fps = fopen(name_s, "r")) != NULL) {
        input(endd);
        loading(fps, name_s, w, f);

        while (ww <= enddd and ff <= enddd and stopp == 0) {
            system("cls");
            pr(A);
            printf("\n");
            human(w, qweqwe, qweqwe1, stop);
            proverka();
            system("cls");
            pr(A);
            if (ww <= enddd and ff <= enddd and stopp == 0){
                mind(n, f, qweqwe, stop);
            }
        }
        if (stopp == 1){
          save(fps, name_s, w, f);
          printf("сохранение");
```

```
        }
        else {
            system("cls");
            end(f, w);
        }
        fclose(fps);
    }
}
```

pr.cpp

```
#include "stdafx.h"
void pr(int* a) {
    printf("    А Б В Г Д Е Ё Ж З И\n");
    for (int i = 0; i < 10; i++) {
        printf(" %d", i + 1);
        if (i < 9) {
            printf(" ");
        }
        for (int j = 0; j < 10; j++) {
            char p1[] = " _", p2[] = " O", p3[] = " M", p4[] = " X";
            int yy = *(a + i*10 + 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 :
                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 D. Схемы функций программы

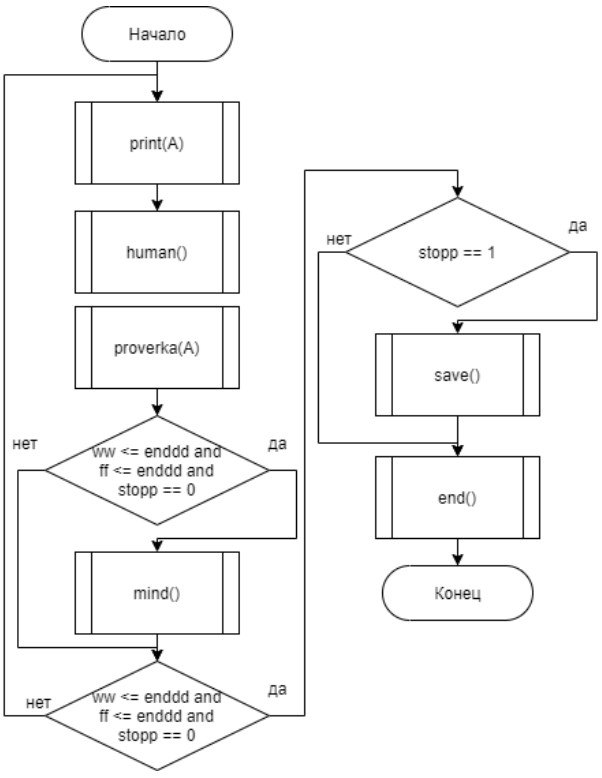

Рисунок 6.1 — Функция main

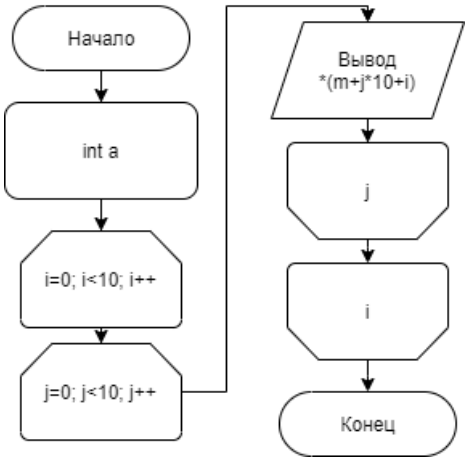

Рисунок 6.2 — Функция pr

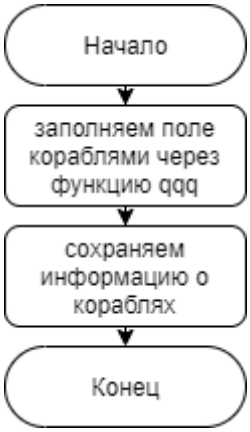

Рисунок 6.3 — Функция input

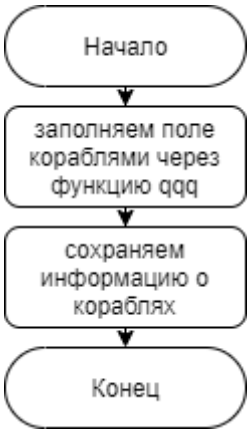

Рисунок 6.4 — Функция qqq

Рисунок 6.5 — Функция proverka

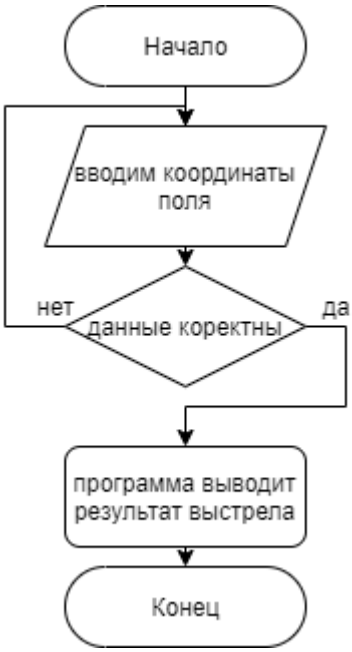

Рисунок 6.6 — Функция human

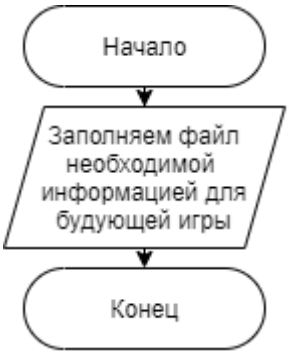

Рисунок 6.7 — Функция save

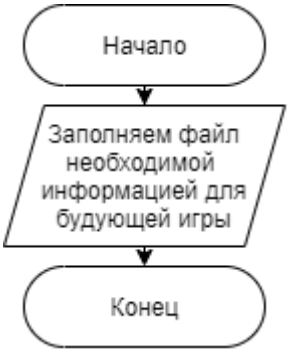

Рисунок 6.8 — Функция loading

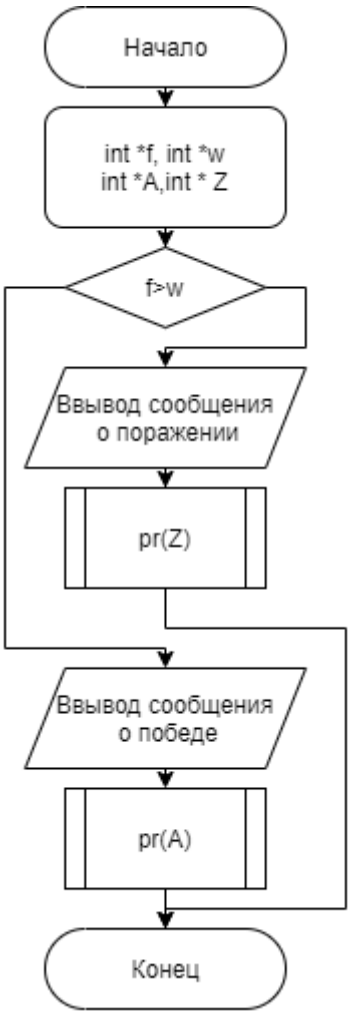

Рисунок 6.9 — Функция end

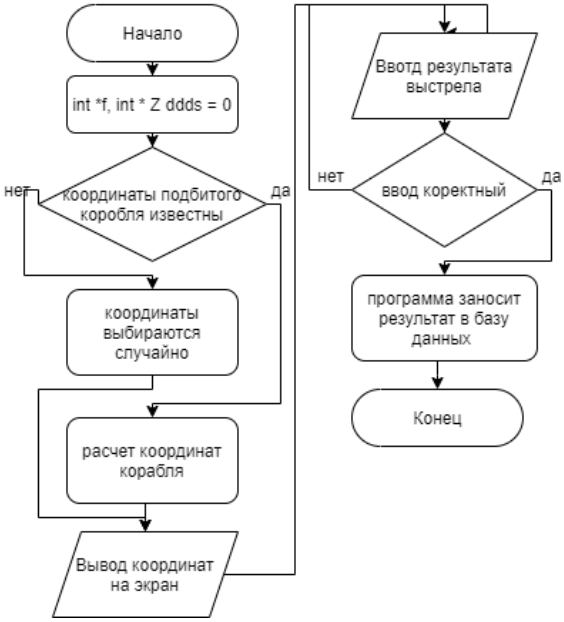

Рисунок 6.9 — Функция mind

Приложение Е. Схема взаимодействия программы

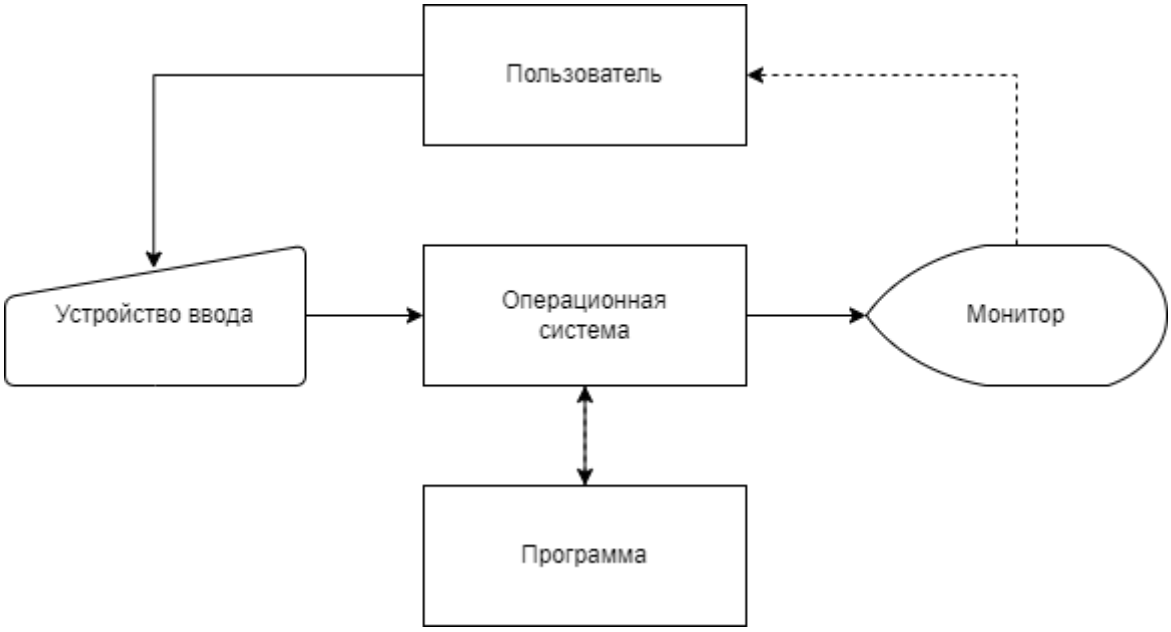

Рисунок 7 — Схема взаимодействия программы