ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
федеральное государственное бюджетное образовательное
учреждение высшего образования
«ОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ»
(ОмГУПС (ОмИИТ))

Кафедра «Автоматика и системы управления»

ПРОГРАММИРОВАНИЕ ИГРЫ НА ЯЗЫКЕ С

Пояснительная записка к курсовой работе
по дисциплине «Программирование»

ИНМВ.400004.000 ПЗ

Студент гр. 21з
_____ Хусаинов К. А.
16.05.2022

Руководитель –
доцент кафедры АиСУ
_____ Пономарев А. В.
«__»_____2022 г.

Омск 2022

Реферат

УДК 004.42

Пояснительная записка к курсовой работе содержит 27 страниц, 16 рисунков, 3 использованных источника, 2 приложения.

Объектом курсовой работы является консольная игра «Норткотта».

Цель курсовой работы – получение основных навыков использования языка Си, создание игры с искусственным интеллектом.

Результатом курсовой работы является игра «Норткотта», написанная на языке Си в программе Visual Studio Code.

В процессе создания игры была изучена лексика, синтаксис и семантика языка Си.

Пояснительная записка выполнена в текстовом редакторе Microsoft Word 2016.

Содержание

Введение

Язык программирования Си – компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969 – 1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашел применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств – от суперкомпьютеров до встраиваемых систем. Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C.

Язык программирования Си отличается минимализмом. Си создавался с одной важной целью: сделать более простым написание больших программ с минимумом ошибок по правилам процедурного программирования, не добавляя на итоговый код программ лишних накладных расходов для компилятора.

Си предлагает следующие важные особенности: простую языковую базу, из которой вынесены в библиотеки многие существенные возможности; ориентацию на процедурное программирование, обеспечивающую удобство применения структурного стиля программирования; систему типов; использование препроцессора; непосредственный доступ к памяти компьютера через использование указателей; минимальное число ключевых слов; передачу параметров в функцию по значению, а не по ссылке; указатели на функции и статические переменные, структуры и объединения; средства объектно-ориентированного программирования.

Часть отсутствующих возможностей относительно легко имитируется встроенными средствами, часть добавляется с помощью сторонних библиотек, часть реализуется в некоторых компиляторах в виде расширений языка.

Язык Си остается языком, реализованным на максимальном количестве аппаратных платформ, и одним из самых популярных языков программирования, особенно в мире свободного программного обеспечения.

Курсовая работа является примером использования языка Си.

1 Правила игры

Норткотта – это логическая игра между двумя игроками, целью каждого из игроков является «закрыть» шашки соперника за неограниченное количество ходов.

В начале игры на поле, произвольного размера, в левом и правом столбце во всех строках располагаются черные и белые шашки. Во время каждого хода, игрок может перемещать одну свою шашку вдоль своего ряда на любое количество клеток. При этом шашка игрока не может «перепрыгивать» через шашку другого, а также, ставить свою шашку на шашку противника – нельзя.

Игроки совершают ходы по очереди. Победу одерживает тот игрок, который своими шашками «закрыл» шашки противника.

При игре против компьютера действуют все те же правила. Игрок произвольно перемещает свою шашку, а компьютер переставляет свою, в зависимости от хода игрока.

Стратегия, как и в любой другой игре тут, конечно же, присутствует. Исходя из моих наблюдений, размышлений и многократных сеансов в этой игре, могу дать определенные советы, позволяющие получить некоторое преимущество над своим соперником:

Если вы ходите первым, то следует «закрывать» шашки соперника до тех пор, пока незакрытых шашек останется 4 штуки. И дальше, перед тем как совершить ход, необходимо тщательно продумать все варианты, для того чтобы не проиграть, то есть не дать сопернику закрыть вашу последнюю незакрытую шашку. Иначе, вы в большинстве случаев потерпите поражение.

Если же вы ходите вторым, то в этом случае следует «закрывать» шашки соперника до тех пор, пока незакрытых шашек останется 6 штук. И дальше, также необходимо просчитывать все варианты событий, чтобы не дать сопернику закрыть вашу последнюю незакрытую шашку.

Если же ваш соперник, как и вы, хорошо осведомлен о выигрышных стратегиях и знает то, как правильно передвигать свои шашки, то игра может затянуться на продолжительное время, а то и вовсе длиться бесконечно. В этой игре не предусмотрен ничейный результат, обязательно должен быть победитель.

И повторюсь, если же кому-то удалось «закрыть шашки» соперника, то этот игрок побеждает. В этом и заключается смысл игры «Норткотта».

2 Реализация игры

В программе используются библиотеки:

– #include <stdio.h> // библиотека ввода–вывода;

– #include <string.h> // библиотека для работы со строками и памятью;

– #include <stdlib.h> // библиотека общего назначения;

– #include <time.h> // библиотека для работы с датой и временем;

– #include <unistd.h> // библиотека для работы с различными процессами.

Программа разработана с использованием структурного подхода к программированию. Для решения поставленной задачи были реализованы несколько функций.

Функции данной игры:

– hello () – функция приветствия;

– rules_of_the_game () – функция для вывода основных правил игры;

– field_size – функция для выбора размера поля;

– init_game – функция для заполнения первоначальных данных;

– game_mode – функция для выбора режима игры и запись имен игроков;

– print_state – функция печати игрового поля;

– print_path – функция печати информации предыдущего хода;

– change_pos – функция сдвига шашек;

– change_pos_PC – функция сдвига шашек для компьютера;

– check_game_over – функция проверки завершения игры;

– open_save_file – функция для открытия файла сохранения;

– show_saves – функция печати слотов для сохранения;

– check_have_save – функция проверки наличия сохранения;

– check_save_file – функция проверки файла сохранения на целостность;

– save_game – функция сохранения текущей игры;

– loading_save_game – функция загрузки сохраненной игры.

Функция hello () ничего не принимает и ничего не возвращает. Выводит на экран фразу «Добро пожаловать в игру «Норткотта»».

Функция rules_of_the_game () ничего не принимает и ничего не возвращает. Выводит на экран правила игры, если того пожелает пользователь.

Функция field_size () ничего не принимает и ничего не возвращает. Дает возможность пользователю выбрать произвольный размер игрового поля. При этом, максимальный размер игрового поля 15 на 15.

Функция init_game () ничего не принимает и ничего не возвращает. Заполняет глобальные переменные первоначальными данными.

Функция game_mode () ничего не принимает и ничего не возвращает. Спрашивает у пользователя количество игроков, которые будут играть, и записывает их имена.

Функция print_state () ничего не принимает и ничего не возвращает. Выводит на экран игровое поле, с учетом изменений, происходящих в игре.

Функция print_path () принимает в качестве аргументов текущего игрока и строку, но ничего не возвращает. Выводит на экран информацию о предыдущем ходе соперника.

Функция change_pos () принимает в качестве аргумента текущего игрока и возвращает текущую строку. Спрашивает у пользователя позицию, в которую он хочет поставить свою шашку, проверяет ее и записывает информацию в переменные.

Функция change_pos_PC () принимает в качестве аргумента сложность игры и возвращает текущую строку. Сдвигает шашку компьютера в зависимости от сложности и ситуации на игровом поле.

Функция check_game_over () ничего не принимает, но возвращает 0 – никто не одержал победу, 1 – кто-то выиграл. Проверяет положение шашек игроков.

Функция open_save_file () принимает в качестве аргументов номер сохранения и номер режима, возвращает 0 – файл не открылся, указатель файла – файл открылся с нужным режимом.

Функция show_saves () принимает в качестве аргумента номер сохранения, но ничего не возвращает. Выводит на экран информацию о слотах сохранений.

Функция check_have_file () принимает в качестве аргументов номер сохранения и номер выполняемой функции, возвращает 0 – слот имеет сохранение и его не стали перезаписывать, 1 – слот имеет сохранение и его перезаписывают, 2 – сохранения не существует, 3 – слот для сохранения свободен. Проверяет наличие сохранения.

Функция check_save_file () принимает в качестве аргумента номер сохранения, возвращает 0 – файл поврежден, 1 – файл в порядке. Проверяет файл сохранения на целостность.

Функция save_game () ничего не принимает, но возвращает 0 – выход без сохранения, 1 – выход с сохранением. Дает возможность сохранить игру в текстовый файл «save1.txt», «save2.txt», «save3.txt», в зависимости от слота сохранения. В один из этих файлов сначала сохраняется длина и ширина поля, количество игроков, текущий игрок, уровень и количество ходов первого и второго игрока. Далее сохраняется положение шашек на игровом поле. И в конце сохраняются имена игроков.

Функция loading_save_game () ничего не принимает, но возвращает 0 – начало новой игры, 1 – загрузка сохраненной игры. Дает возможность загрузить сохраненную игру.

Глобальные переменные данной игры:

– field_row – длина поля;

– field_col – ширина поля;

– players_count – количество игроков;

– level – сложность;

– gamer – текущий игрок;

– field[15][15] – игровое поле;

– name1, name2 – имена игроков;

– pos1[15], pos2[15] – положение шашек по горизонтали;

– pos1_old[15], pos2_old[15] – старое положение шашек по горизонтали;

– moves_1, moves_2 – количество ходов первого и второго игрока.

3 Инструкция пользователя

При запуске игры, после приветствия нам предлагают ознакомиться с правилами игры, мы можем согласиться, нажав «Y» или «y» (рисунок 1).

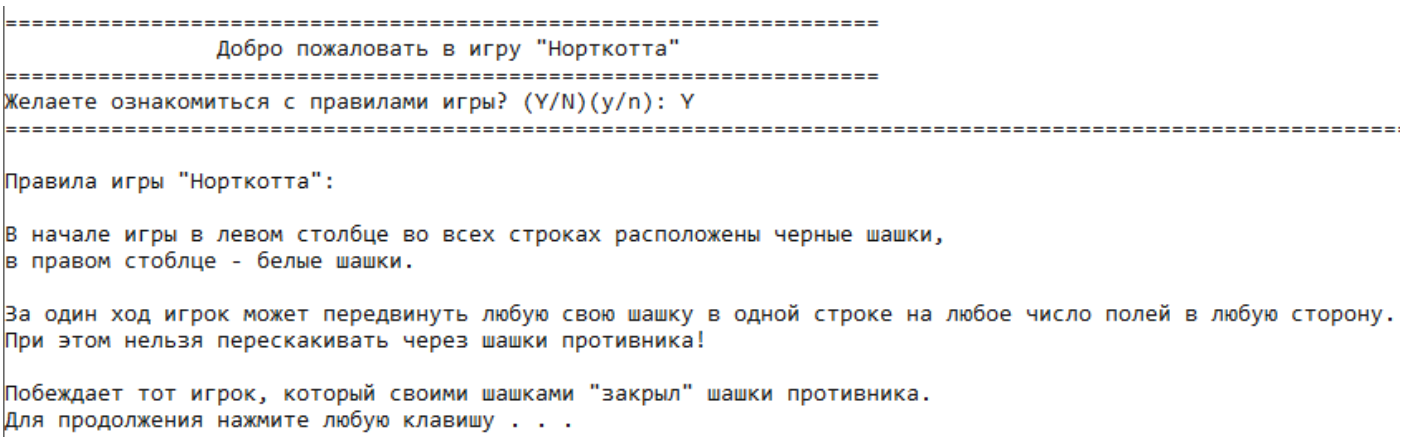

Рисунок 1 – Согласие на ознакомление с правилами игры

А также можем не ознакомляться с правилами (рисунок 2).

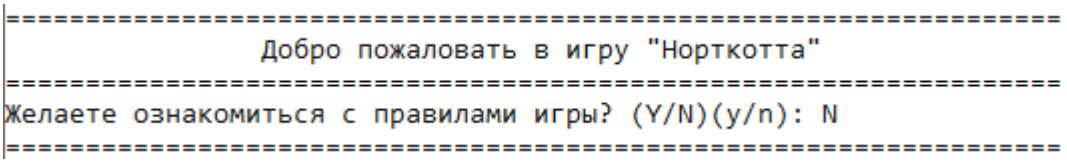

Рисунок 2 – Отказ от ознакомлений с правилами

После правил игры, нам предлагают загрузить сохраненную игру, мы может согласиться, нажав «Y» или «y», а после посмотреть имеющиеся сохранения, нажав «0» (рисунок 3).

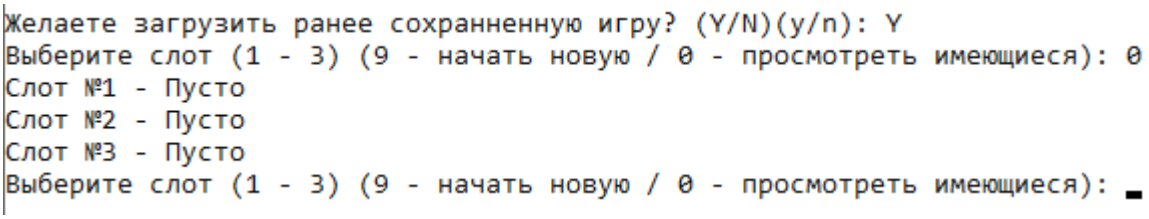

Рисунок 3 – Меню загрузки сохраненной игры

Нажмем «9», то есть начнем новую игру, и мы увидим, как нам предлагают выбрать размер поля, введем «5*8» (рисунок 4).

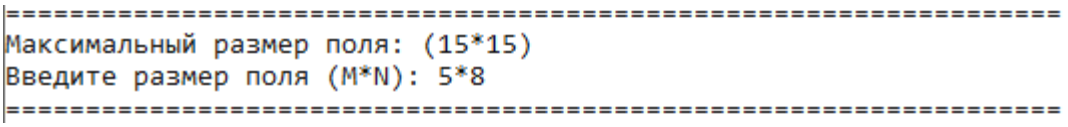

Рисунок 4 – Выбор размера поля

Попробуем ввести недопустимые размеры поля (рисунок 5).

```
Максимальный размер поля: (15*15)
Введите размер поля (M*N): 16*15
Некорректный размер поля, повторите ввод
Введите размер поля (M*N): 15*16
Некорректный размер поля, повторите ввод
Введите размер поля (M*N): 1*1
Некорректный размер поля, повторите ввод
Введите размер поля (M*N): 0*0
Некорректный размер поля, повторите ввод
Введите размер поля (M*N): 1*2
Некорректный размер поля, повторите ввод
```

Рисунок 5 – Ввод недопустимых размеров поля

После выбора размера поля, нам предлагается выбрать режим игры, то есть игра с компьютером (1) или с пользователем (2), выберем 2 (рисунок 6).

```
Сколько игроков будет играть? (1 или 2)
Введите количество игроков: 2
```

Рисунок 6 – Выбор режима игры

После выбора режима, нам предлагается записать имена игроков, запишем произвольные имена без пробелов, иначе запишется только первая часть имени (рисунок 7).

```
Сколько игроков будет играть? (1 или 2)
Введите количество игроков: 2
Введите имя первого игрока: User-1
Введите имя второго игрока: User-2
=====================================================================
```

Рисунок 7 – Запись имен пользователей

После записи имен, появляется поле и для того чтобы переставить шашку на выбранную ячейку, необходимо ввести координаты через пробел, например, введем «1 6» (рисунок 8).

```
User-1 (Слева)   vs   User-2 (Справа)

     |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   1 |  .  |  .  |  .  |  .  |  .  |  O  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   2 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   3 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   4 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   5 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|

Предыдущий ход игрока №1 - User-1: Ряд №1 (1 ---> 6)
-----------------------------------------------------
User-2, введите позицию, в которую хотите поставить шашку (0 0 - выйти из игры):
```

Рисунок 8 – Сдвиг шашки

Попробуем поставить шашку второго игрока против правил игры (рисунок 9).

```
User-1 (Слева)   vs   User-2 (Справа)

     |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   1 |  .  |  .  |  .  |  .  |  .  |  O  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   2 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   3 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   4 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   5 |  O  |  .  |  .  |  .  |  .  |  .  |  .  |  X  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|

Предыдущий ход игрока №1 - User-1: Ряд №1 (1 ---> 6)
--------------------------------------------------------
User-2, введите позицию, в которую хотите поставить шашку (0 0 - выйти из игры): 1 5
Некорректная позиция.
User-2, введите позицию, в которую хотите поставить шашку (0 0 - выйти из игры): 1 6
Некорректная позиция.
User-2, введите позицию, в которую хотите поставить шашку (0 0 - выйти из игры):
```

Рисунок 9 – Ввод некорректных ячеек

После того, как один из игроков выиграл, то можно увидеть результат (рисунок 10).

```
User-1 (Слева)   vs   User-2 (Справа)

     |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   1 |  O  |  X  |  .  |  .  |  .  |  .  |  .  |  .  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   2 |  O  |  X  |  .  |  .  |  .  |  .  |  .  |  .  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   3 |  O  |  X  |  .  |  .  |  .  |  .  |  .  |  .  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   4 |  O  |  X  |  .  |  .  |  .  |  .  |  .  |  .  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
   5 |  O  |  X  |  .  |  .  |  .  |  .  |  .  |  .  |
-----|-----|-----|-----|-----|-----|-----|-----|-----|

Предыдущий ход игрока №2 - User-2: Ряд №1 (7 ---> 2)
--------------------------------------------------------
Игра окончена!
============================================================
За 6 ходов победу одержал игрок №2 - User-2.
Для продолжения нажмите любую клавишу . . .
```

Рисунок 10 – Итог

Заключение

Во время выполнения курсовой работы были изучены и разобраны такие элементы языка программирования Си как, библиотеки, способные добавить разрабатываемому приложению функциональности, переменные, типы данных, циклы, операторы и функции.

На каждом этапе разработки данной игры у нас возникали небольшие проблемы, которые нам приходилось разрешать. Например, на начальном этапе, мы не знали где и как хранить данные, необходимые для работоспособности игры. Но изучив дополнительные материалы, и освежив в памяти некоторые знания, мы успешно разрешили эту проблему. Далее, у нас появился вопрос, как реализовать игру против компьютера. Попробовав все возможные варианты реализации, мы нашли, по нашему мнению, самый оптимальный и остановились на нем. И на заключительном этапе, у нас вызвало затруднение работа с текстовыми файлами. Изучив дополнительную информацию в литературе и на просторах интернета, мы нашли выход из этой ситуации и благополучно реализовали задуманное. В результате чего у нас получилась консольная игра «Норткотта», которая является результатом выполнения данной курсовой работы.

Библиографический список

1 Википедия. Свободная энциклопедия [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/Си_(язык_программирования)

2 Основы языка С: Методические указания к лабораторным работам / Е. А. Альтман, А. В. Александров, Н. Г. Ананьева, Н. Е. Актаев; Омский гос. ун-т путей сообщения. Омск, 2012. 46 с.

3 Введение в программирование: Методические указания к лабораторным работам / Е. А. Альтман, А. В. Александров, Н. Г. Ананьева, Н. Е. Актаев; Омский гос. ун-т путей сообщения. Омск, 2011. 31 с.

Приложение А
(обязательное)
Код программы

```
#include<stdio.h>               // Библиотека для ввода и вывода
#include<string.h>              // Библиотека для функции strcpy()
#include<stdlib.h>              // Библиотека для функции rand(), srand()
#include<time.h>                // Библиотека для функции time()
#include<unistd.h>              // Библиотека для функции sleep()
int field_row;                  // Длина поля
int field_col;                  // Ширина поля
int players_count;              // Количество игроков
int level;                      // Сложность
int gamer=1;                    // Текущий игрок (первоначально 1-ый)
char field[15][15];             // Поле размером 15 на 15
char name1[15], name2[15];      // Имена игроков по 15 символов
int pos1[15], pos2[15];         // Позиция шашек игроков по горизонтали
int moves_1, moves_2;           // Количество ходов, который совершил каждый
игрок
int pos1_old[15], pos2_old[15]; // Старые позиции шашек игроков по гориз.
/*  Объявление функций  */
void hello();
void rules_of_the_game();
void field_size();
void init_game();
void game_mode();
void print_state();
void print_path(int gamer, int row);
int change_pos(int gamer);
int change_pos_PC(int level);
int check_game_over();
FILE *open_save_file(int save_number, int mode);
void show_saves(int save_number);
int check_have_save(int save_number, int func);
int check_save_file(int save_number);
int save_game();
int loading_save_game();

/*  Функции: */
// Приветствие
void hello(void){
printf("=================================================================\n");
    printf("\t\tДобро пожаловать в игру \"Норткотта\"\n");

printf("=================================================================\n");
}
```

Листинг А.1 – Исходный код файла game.c, лист 1

```
// Правила игры
void rules_of_the_game(void){
    char c;
    printf("Желаете ознакомиться с правилами игры? (Y/N)(y/n): ");
    while(1){
        scanf("%c", &c);
        if(c=='Y' || c=='y'){
            printf("=====================================================");
            printf("\nПравила игры \"Норткотта\":\n\n");
            printf("В начале игры в левом столбце во всех строках расположены ");
            "черные шашки,\n");
            printf("в правом столбце - белые шашки.\n\n");
            printf("За один ход игрок может передвинуть любую свою шашку ");
            "в одной строке на любое число полей в любую сторону.\n");
            printf("При этом нельзя перескакивать через шашки противника!\n\n");
            printf("Побеждает тот игрок, ");
            "который своими шашками \"закрыл\" шашки противника.\n");
            system("Pause");
            printf("=====================================================");
            "=========================================================\n");
            break;
        }
        else if(c=='N' || c=='n'){
            printf("================================");
            "==================================\n");
            break;
        }
    }
}
// Выбор размера поля
void field_size(void){
    printf("Максимальный размер поля: (15*15)\n");
    while(1){
        printf("Введите размер поля (M*N): ");
        scanf("%d*%d", &field_row, &field_col);
        if(field_row>15 || field_col>15 || field_col<3 || field_row<1)
            printf("Некорректный размер поля, повторите ввод\n");
        else{
            printf("================================");
            "==================================\n");
            break;
        }
    }
}
```

Листинг А.1, лист 2

```
// Заполнение начальными данными
void init_game(void){
    int i;
    for(i=0;i<field_row;i++){
        field[i][0]=1;           // Белые шашки
        field[i][field_col-1]=2; // Черные шашки
        pos1[i]=0;               // Первоначальные позиции шашек 1-го игрока по
горизонтали
        pos2[i]=field_col-1;     // Первоначальные позиции шашек 2-го игрока по
горизонтали
        pos1_old[i]=0;           // Старые позиции шашек 1-го игрока по
горизонтали
        pos2_old[i]=field_col-1; // Старые позиции шашек 2-го игрока по
горизонтали
    }
    moves_1=0;                   // Количество ходов совершенным 1-ым игроком
    moves_2=0;                   // Количество ходов совершенным 2-ым игроком

printf("=================================================================\n");
    printf("Идет загрузка...");
    sleep(1);
}
// Выбор режима игры (1 игрок/2 игрока), запись имен
void game_mode(void){
    printf("Сколько игроков будет играть? (1 или 2)\n");
    while(1){
        printf("Введите количество игроков: ");
        scanf("%d", &players_count);
        if(players_count==2){ // Если будут играть 2 игрока
            printf("Введите имя первого игрока: ");
            scanf("%s", name1);
            printf("Введите имя второго игрока: ");
            scanf("%s", name2);
            break;
        }
        else if(players_count==1){ // Если 1...
           printf("Введите имя игрока: ");
           scanf("%s", name1);
           strcpy(name2, "Компьютер");
           printf("Выберите сложность (1 - легко / 2 - средне / 3 - сложно): ");
           scanf("%d", &level);
           break;
        }
        else{
            printf("Некорректное число игроков. Повторите ввод.\n\n");
        }
    }
}
```

Листинг А.1, лист 3

```
// Печать игрового поля
void print_state(void){
    int i,j,k;
    printf("\n%s (Слева)   vs   %s (Справа)\n\n", name1, name2); // Печать имен
    printf("     | ");
    for(i=0;i<field_col;i++)
        printf("%2d  | ", i+1); // Печать координат по горизонтали
    printf("\n");
    printf("-----|");
    for(i=0;i<field_col;i++)
        printf("-----|");
    printf("\n");
    for(i=0; i<field_row; i++){
        printf("%4d ", i+1); // Печать координат по вертикали
        for(j=0; j<field_col; j++)
            if(field[i][j]==0){
                printf("|  .  ");  // Печать пустых ячеек
            }
            else{
                field[i][j]==1 ? printf("|  O  ") : printf("|  X  "); // Печать шашек
            }
        printf("|");
        printf("\n");
        for(k=0;k<=field_col;k++)
            printf("-----|");
        printf("\n");
    }
}
// Печать пути прошлого хода
void print_path(int gamer, int row){
    if(gamer==1){ // Если ход совершает 1-ый игрок, то выводится информация о 2-ом
        printf("\nПредыдущий ход игрока №2 - %s: Ряд №%d (%d ---> %d)\n", name2, row,
(pos2_old[row-1]+1), (pos2[row-1]+1));
        printf("-------------------------------------------------------");
    }
    else{ // Если 2-ой...
        printf("\nПредыдущий ход игрока №1 - %s: Ряд №%d (%d ---> %d)\n", name1, row,
(pos1_old[row-1]+1), (pos1[row-1]+1));
        printf("-------------------------------------------------------");
    }
}
```

Листинг А.1, лист 4

```
// Сдвиг шашек и проверка (для пользователей)
int change_pos(int gamer){
    int row, col;
    while(1){
      if(gamer==1){
          printf("\n%s, ", name1);
      }
      else{
          printf("\n%s, ", name2);
      }
      printf("введите позицию, в которую хотите поставить шашку (0 0 - выйти из игры):");
      scanf("%d %d", &row, &col);
       if(row==0 && col==0){ // Если пользователь хочет выйти
       printf("================================================================================
       ===========\n");
          return 0;
      }
        row=row-1;
        col=col-1;
        if(row>=0 && row<field_row && col>=0 && col<field_col) // Проверка диапазона
            if(field[row][col]==0 && gamer==1 && col<pos2[row]){ // Если ячейка не
занята, игрок №1, и не переходит за другую шашку
                field[row][pos1[row]]=0; // Убираем шашку со старой позиции
                field[row][col]=1; // Ставим шашку в новую позицию
                pos1_old[row]=pos1[row];
                pos1[row]=col; // Фиксируем новое положение по горизонтали
                system("cls");
                break;
            }
            else if(field[row][col]==0 && gamer==2 && pos1[row]<col){ // Если ячейка не
занята, игрок №2, и не переходит за другую шашку
                field[row][pos2[row]]=0; // Убираем шашку со старой ячейки
                field[row][col]=2; // Ставим шашку в новую ячейку
                pos2_old[row]=pos2[row]; // Записываем старое положение
                pos2[row]=col; // Записываем новое положение
                system("cls");
                break;
            }
            else
                printf("Некорректная позиция.");
    }
    if(gamer==1) // Если ход совершил 1-ый игрок, то увеличиваем кол-во ходов 1-го игрока
        moves_1++;
    else // Если 2-ой...
        moves_2++;
    return row+1;
}
```

Листинг А.1, лист 5

```
// Сдвиг шашек и проверка (для компьютера)
int change_pos_PC(int level){
    int row, col, i, count=0, try=0;
    for(i=0;i<field_row;i++
        if(pos2[i]==pos1[i]+1)
            count++; // Количество шашек, которые не могут ходить вперед
        }
    }
    srand(time(NULL));
    while(1){
        row=0+rand()%((field_row-1)-0+1);
        if(level==1){ // Легко
            col=1+rand()%((field_col-1)-1+1);
        }
        else if(level==2){ // Средне
            col=(pos1[row]+1)+rand()%((pos1[row]+3)-(pos1[row]+1)+1);
            if(count==field_row){
                col=(pos2[row]+1)+rand()%((pos2[row]+3)-(pos2[row]+1)+1);
            }
        }
        else if(level==3){ // Сложно
            col=pos1[row]+1;
            if(count==(field_row-3)){
                col=pos1[row]+1;
            }
            if(count==(field_row-2)){
                col=(pos1[row]+2)+rand()%((pos1[row]+4)-(pos1[row]+2)+1);
                if(try==(field_row+25)){
                    col=pos2[row]-1;
                    try=0;
                }
                if(col>=field_col){
                    try++;
                }
                if(col==pos2[row]){
                    col=pos2[row]+1;
                }
            }
            else if(count==(field_row-1)){
                col=pos1[row]+1;
            }
            else if(count==field_row){
                col=(pos2[row]+1)+rand()%((pos2[row]+3)-(pos2[row]+1)+1);
            }
        }
```

Листинг А.1, лист 6

```
if(row>=0 && row<field_row && col>=0 && col<field_col){ // Проверка диапазона
            if(field[row][col]==0 && pos1[row]<col){ // Если ячейка свободна и
не "переходит" за другую шашку
                field[row][pos2[row]]=0; // Убираем шашку со старой ячейки
                field[row][col]=2; // Ставим шашку в новую ячейку
                pos2_old[row]=pos2[row]; // Записываем старое положение
                pos2[row]=col; // Записываем новое положение
                moves_2++; // Увеличиваем ходы компьютера
                sleep(1);
                system("cls");
                break;
            }
        }
    }
    return (row+1);
}
/*
    Проверка завершения игры
    0 - Победу никто не одержал
    1 - Победу одержал один из игроков
*/
int check_game_over(void){
    int i, count_1=0, count_2=0;
    for(i=0;i<field_row;i++){
        if(field[i][field_col-2]==1){ // Если шашка первого игрока "перекрыла"
шашку второго игрока
            count_1++;
        }
        else if(field[i][1]==2){ // Если шашка второго игрока "перекрыла" шашку
первого игрока
            count_2++;
        }
    }
    if(count_1==field_row || count_2==field_row){
        printf("\nИгра окончена!\n");

printf("=====================================================================\n");
        if(count_1==field_row){
            printf("За %d ходов победу одержал %s.\n",moves_1, name1);
        }
        else{
            printf("За %d ходов победу одержал %s.\n",moves_2, name2);
        }
        return 1;
    }
    return 0;
}
```

Листинг А.1, лист 7

```
/*
    Открывает файл сохранения для определенной задачи:
    0 - Файл не открылся
    f - Файл открылся
    mode (1 - чтение, 2 - запись)
*/
FILE *open_save_file(int save_number, int mode){
    FILE *f;
    char namefile[10]={"save0.txt"};
    namefile[4]+=save_number;
    if(mode==1){
        f=fopen(namefile, "r");
    }
    else{
        f=fopen(namefile, "w");
    }
    if(!f){ // Если не открылся, то возвращаем 0
        return NULL;
    }
    else{ // Если открылся, то возвращаем указатель на файл
        return f;
    }
}
// Отображение слотов
void show_saves(int save_number){
    FILE* f;
    f=open_save_file(save_number, 1);
    if(f){ // Если файл открылся, то он существует
        printf("Слот №%d - Занят\n", save_number);
        fclose(f);
    }
    else{ // Файл не существует
        printf("Слот №%d - Свободен\n", save_number);
    }
}
/*
    Проверка наличия сохранения:
    0 - Файл уже имеет сохранение и его НЕ перезаписывают
    1 - Файл уже имеет сохранение и его перезаписывают
    2 - Сохранения не существует
    3 - Слот для сохранения свободен
    func (1 - сохранение, 2 - загрузка)
*/
```

Листинг А.1, лист 8

```
int check_have_save(int save_number, int func){
    char symbol;
    FILE* f;
    f=open_save_file(save_number, 1);
    if(f && func==1){ // Если файл открылся и его хотят сохранить
        printf("В данном слоте уже имеется сохранение.\nВы хотите перезаписать
сохранение? (Y/N)(y/n): ");
        while(1){
            scanf("%c", &symbol);
            if(symbol=='Y' || symbol=='y'){
                return 1;
            }
            else if(symbol=='N' || symbol=='n')
                return 0;
        }
    }
    else if(!f && func==1) // Если файл не открылся и его хотят сохранить, то
слот свободен
        return 3;
    else if(!f && func==2){ // Если файл не открылся и его хотят загрузить, то
его не существ.
        printf("Сохранения №%d не существует! Выберите другой слот.\n",
save_number);
        return 2;
    }
}
/*
    Проверка файла на целостность:
    0 - Файл поврежден
    1 - Файл целостен
*/
int check_save_file(int save_number){
    char str[15];
    int line=0;
    FILE* f;
    f=open_save_file(save_number, 1);
    fscanf(f,"%d ", &field_row);
    while(fgets(str,15,f)) // Пока не достигнут конец файла считываем строки
        line++;
    if(line-1!=(2+(4*field_row))){ // Если количество строк, которое должно
быть, не совпадает с текущим
        printf("Файл сохранения №%d поврежден. Удалите и выберите другой.\n",
save_number);
        field_row=0;
        return 0;
    }
    else
        return 1;
}
```

Листинг А.1, лист 9

```
/*  Сохранение текущей игры:
    0 - Выход из игры без сохранения
    1 - Выход из игры с сохранением    */
int save_game(void){
    char symbol;
    int save_number, i, j, step;
    printf("Желаете сохранить игру? (Y/N)(y/n): ");
    while(1){
        scanf("%c", &symbol);
        if(symbol=='N' || symbol=='n'){
printf("================================================================================
====\n");
            return 0;
        }
        else if(symbol=='Y' || symbol=='y')
            break;
    }
    while(1){
        printf("Выберите слот (1 - 3) (9 - выйти / 0 - просмотреть сохранения): ");
        scanf("%d", &save_number);
        if(save_number==9){

printf("================================================================================
====\n");
            return 0;
        }
        if(save_number==0)
            for(i=1;i<=3;i++)
                show_saves(i); // Просмотр имеющихся сохранений
            continue;
        FILE* f;
        step=check_have_save(save_number, 1); // Проверка файла на наличие в нем уже
сохранения
        if(!step) // Если сохранение не стали перезаписывать - выбираем другой слот
            continue;
        f=open_save_file(save_number, 2);
        fprintf(f,"%d %d %d %d %d %d %d\n", field_row, field_col, players_count, gamer,
level, moves_1, moves_2);
        for(i=0;i<field_row;i++)
            fprintf(f,"%d\n%d\n%d\n%d\n", pos1[i], pos1_old[i], pos2[i], pos2_old[i]);
        fprintf(f,"%s %s", name1, name2);
        break;
    }
    printf("\nИдет сохранение...");
    sleep(1);
    printf("\nИгра сохранена.\n\n");
    printf("================================================================\n");
    return 1;
}
```

Листинг А.1, лист 10

```
 /* Загрузка ранее сохранненной игры:
    0 - Начало новой игры / 1 - Загрузка сохраненной игры */
int loading_save_game(void){
    char symbol;
    int save_number, i, step;
    printf("Желаете загрузить ранее сохранненную игру? (Y/N)(y/n): ");
    while(1){
        scanf("%c", &symbol);
        if(symbol=='N' || symbol=='n'){

printf("=====================================================================\n");
            return 0;
        }
        else if(symbol=='Y' || symbol=='y')
            break;
    }
    while(1){
        printf("Выберите слот (1 - 3) (9 - начать новую / 0 - просмотреть имеющиеся): ");
        scanf("%d", &save_number);
        if(save_number==9){

printf("=====================================================================\n");
            return 0;
        }
        if(save_number==0)
            for(i=1;i<=3;i++)
                show_saves(i); // Просмотр имеющихся сохранений
            continue;
        FILE* f;
        step=check_have_save(save_number, 2); // Проверка на наличие сохранения
        if(step==2) // Если сохранения не существует - выбираем другое
            continue;
        step=check_save_file(save_number); // Проверка файла на целостность
        if(!step) // Если поврежден - выбираем другой
            continue;
        f=open_save_file(save_number, 1);
        fscanf(f,"%d %d %d %d %d %d %d\n", &field_row, &field_col, &players_count,
&gamer, &level, &moves_1, &moves_2);
        for(i=0;i<field_row;i++){
            fscanf(f,"%d\n%d\n%d\n%d\n", &pos1[i], &pos1_old[i], &pos2[i], &pos2_old[i]);
            field[i][pos1[i]]=1;
            field[i][pos2[i]]=2;
        }
        fscanf(f,"%s %s", &name1, &name2);
        break;
    }
    sleep(1);
    return 1;
}
```

Листинг А.1, лист 11

```
void main(){
    int step, row;
    hello();
    rules_of_the_game();
    step=loading_save_game();
    if(step==0){ // Если начинаем новую игру
        field_size();
        game_mode();
        init_game();
    }
    system("cls");
    print_state();
    if(players_count==2){ // Цикл для двух игроков
        while(1){
            step=check_game_over();
            if(step) // Выход из игры если кто-то победил
                break;
            row=change_pos(gamer);
            if(!row){ // Выход из игры по инициативе пользователя
                save_game();
                break;
            }
            print_state();
            gamer=3-gamer; // Смена игрока
            print_path(gamer, row);
        }
    }
    else { // Цикл для игры с компьютером
        while(1){
            row=change_pos(gamer);
            if(!row){ // Выход из игры по инициативе пользователя
                save_game();
                break;
            }
            print_state();
            step=check_game_over();
            if(step) // Выход из игры если кто-то победил
                break;
            printf("\nКомпьютер думает...");
            sleep(1);
            row=change_pos_PC(level);
            print_state();
            print_path(gamer, row);
            step=check_game_over();
            if(step) // Выход из игры если кто-то победил
                break;
        }
    }
}
```

Листинг А.1, лист 12

Приложение Б
(обязательное)
Пример работы программы

Начинаем игру (рисунок Б.1).

```
======================================================================
                Добро пожаловать в игру "Норткотта"
======================================================================
```

Рисунок Б.1 – Приветствие

Ознакомимся с правилами игры (рисунок Б.2).

```
Желаете ознакомиться с правилами игры? (Y/N)(y/n): y
======================================================================================================
Правила игры "Норткотта":

В начале игры в левом столбце во всех строках расположены черные шашки,
в правом стоблце - белые шашки.

За один ход игрок может передвинуть любую свою шашку в одной строке на любое число полей в любую сторону.
При этом нельзя перескакивать через шашки противника!

Побеждает тот игрок, который своими шашками "закрыл" шашки противника.
Для продолжения нажмите любую клавишу . . .
```

Рисунок Б.2 – Правила игры

Выберем размер поля, режим игры и введем имя (рисунок Б.3).

```
Желаете загрузить ранее сохранненную игру? (Y/N)(y/n): N
======================================================================
Максимальный размер поля: (15*15)
Введите размер поля (M*N): 6*7
======================================================================
Сколько игроков будет играть? (1 или 2)
Введите количество игроков: 1
Введите имя игрока: Карим
```

Рисунок Б.3 – Выбор размера поля, режима игры, ввод имени

Выберем сложность (рисунок Б.4).

```
Выберите сложность (1 - легко / 2 - средне / 3 - сложно): 3
======================================================================
Идет загрузка...
```

Рисунок Б.4 – Выбор сложности

После того, как появилось игровое поле, выполним произвольные ходы (рисунок Б.5).

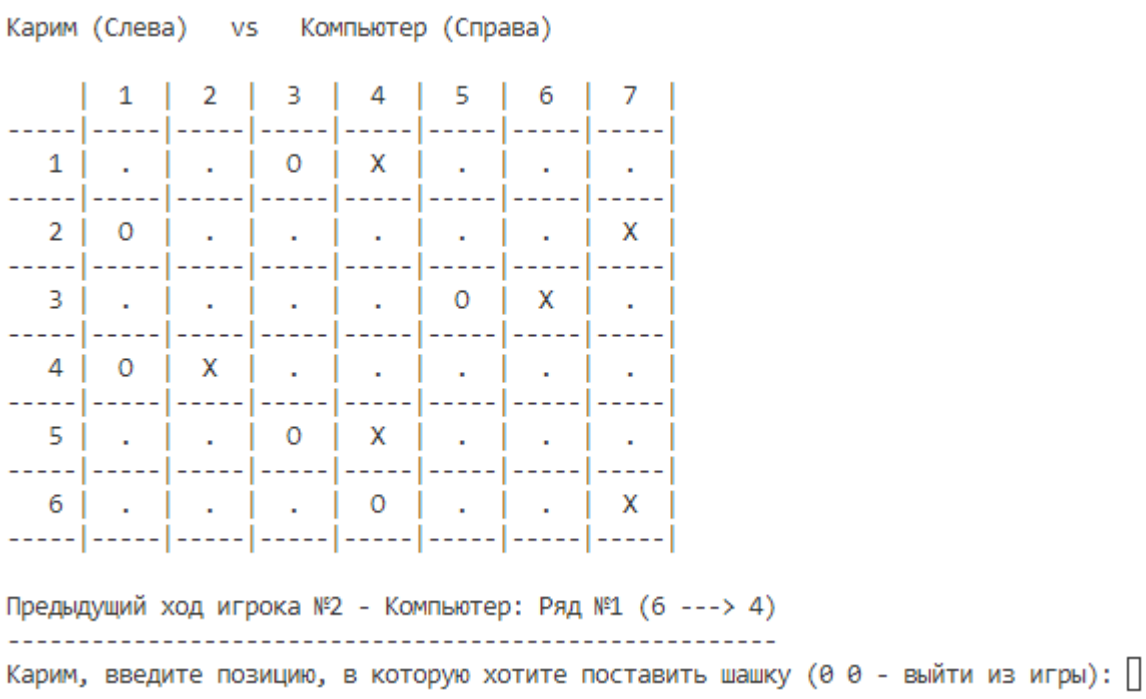

Рисунок Б.5 – Игровое поле

Спустя несколько ходов, мы видим итог (рисунок Б.6).

Рисунок Б.6 – Итог