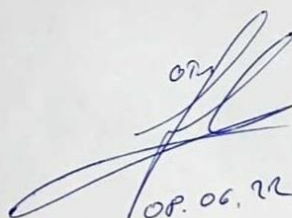


Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

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

Выполнил:
студент группы 21ВВ3
Юсеев Р.Р



08.06.22

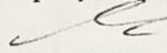
Принял:
аспирант Патушин Д.В.
зам.Декана Голотенков Н.О

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 Юсееву Руслану Лиматовичу Группа 21 ВВЗ

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

Исходные данные (технические требования) на проектирование

Примечание: игра в крестики-нолики.

*Функционал игры: построение игрового поля
автоматический ход компьютера
сохранение результатов игры в файл с указанием имени игрока
вывод лучших пяти игроков по запросу*

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" февраля

Дата защиты проекта "06" 06.22

Руководитель Коротешков Николай Олегович

Задание получил "25" февраля 2022г.

Студент Исеев Сулган Сулганович

Содержание

Постановка задачи	5
Описание структуры программы	6
Описание функционала программы	10
Руководство пользователя	11
Разработка функции на языке Assembler	15
Заключение	16
Список используемых источников	17
Приложение А. Листинги программы	18
Приложение В. Результат работы программы	25

Постановка задачи

Необходимо разработать приложение «Игра в крестики-нолики». Приложение должно строить игровое поле, сохранять результаты игры в файл с указанием имени игрока, выводить лучших пяти игроков по запросу, а также обеспечивать автоматический ход компьютера.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы.

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

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

Устройство ввода информации – клавиатура, мышь. Необходимо определиться с механикой взаимодействия игрока с игрой и её интерфейсом

Пользовательский интерфейс должен быть построен на основе меню и панели инструментов. Это необходимо для создания интуитивно понятного интерфейса для пользователя.

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

Язык программирования: Си и Ассемблер.

Описание структуры программы

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

- 1) TicTacToe.cpp – основной файл программы. Меню программы.
- 2) GameField.cpp – файл создания и обнуления игрового поля. Состоит из функций *visual* – визуализация игрового поля, *nul* обнуляет игровое поле.
- 3) PlayerMove.cpp – файл взаимодействия игрока с игровым полем.
- 4) AIMove.cpp – файл взаимодействия компьютера с игровым полем.
- 5) CheckStat.cpp – файл проверки состояния игры.
- 6) Result.cpp – файл вывода результатов игры в файл.
- 7) TopList.cpp – файл формирования списка пяти лучших игроков. Состоит из следующих функций: *CreateNode* – создание узла списка, *AddNode* – добавление узла в список, *SortList* – сортировка списка, *FormList* – формирование списка, *List* - просмотр списка.

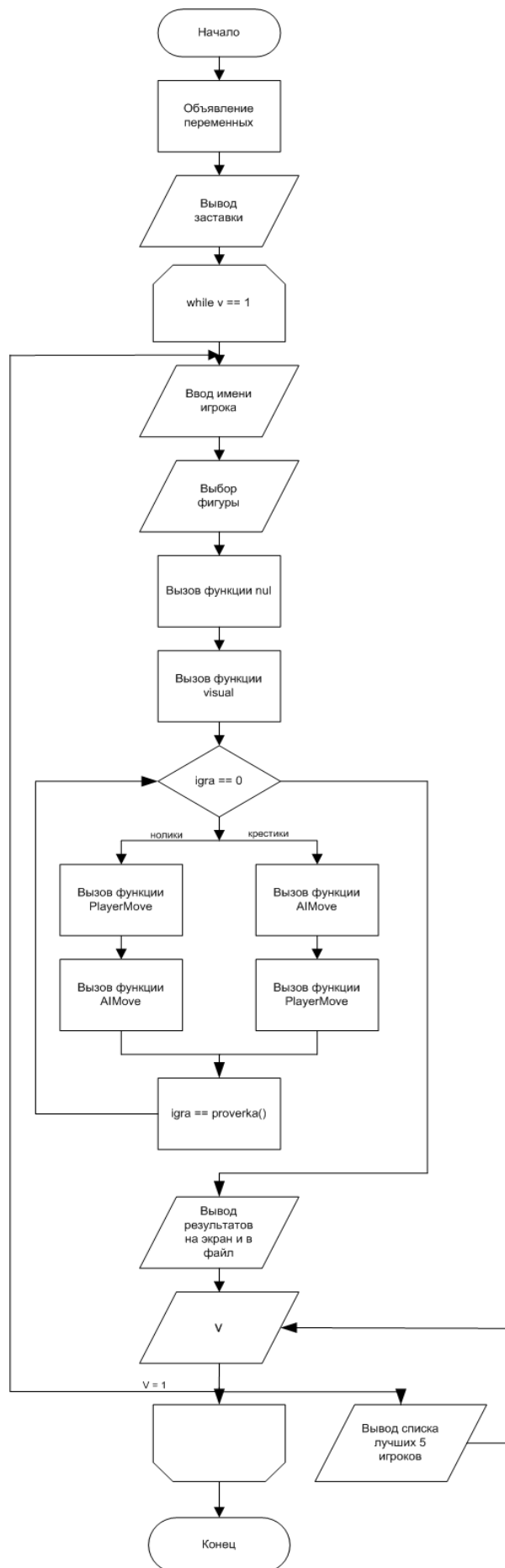


Рисунок 1. Схема функции main

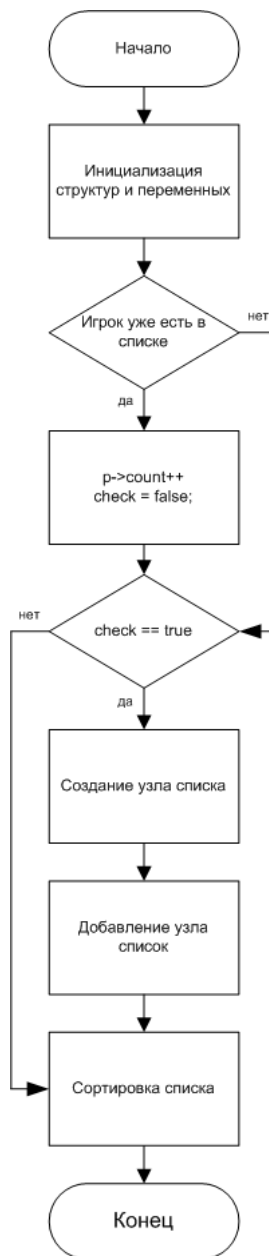


Рисунок 2. Структура схемы FormList

Описание функционала программы

Функция *main* – главная функция. В ней реализуется весь игровой процесс. Сначала выводится заставка. Она сообщает игроку информацию о создателе игры.

Потом программа просит игрока ввести имя. После этого выводится меню с выбором фигуры. Нажав 1, игрок начинает игру ноликами, а нажав 2 – крестиками. Вызываются функции *nul* и *visual* и выводится игровое поле.

Игроку предлагают ввести номер строки и столбца. После ввода номера строки и столбца вызывается функция *PlayerMove*, которая в соответствии с выбранным номером и фигурой выставляет нужную фигуру на клетку. Если клетка уже занята, то программа не дает сделать некорректный ход.

Далее свой ход делает компьютер. Вызывается функция *AIMove*, в которой выбирается оптимальный для компьютера ход.

Игра продолжается до момента, пока один из игроков не составит выигрышную комбинацию, либо же пока на поле не останется свободного места.

После каждого хода функция *CheckStat* проверяет состояние игры: выиграли кто-либо или же игра продолжается. Если функция обнаружила, что игра окончена, программа выводит результаты игры на экран. Если игрок выиграл, его имя и количество побед заносится в список игроков (функция *FormList*).

После каждой игры вызывается функция *result_to_file*, которая заносит результаты в отдельный файл.

В конце игры игроку выводится меню: предлагается сыграть ещё раз, окончить игру, либо же посмотреть список лучших пяти игроков (функция *List*).

Для игрока и противника создается игровое поле – матрица размером 3 на 3. Значение элемента матрицы соответствует поставленной фигуре: 0 – фигуры нет, 1 – поставлен нолик, 2 – поставлен крестик.

Переменные *igra* и *res* отвечают за состояние игры. Если эти переменные равны нулю, то никто не победил, если 1 – победили нолики, если 2 – крестики, если 3 – ничья. В переменную *hi* записывается ход игрока, например, «12» – первая цифра отвечает за строку, вторая цифра за столбец игрового поля.

Переменные *str* и *sto* отвечают за номер выбранных строки и столбца. В переменную *osch* записывается номер игрока, делающего текущий ход, 1 – нолик, 2 – крестик. Переменная *v* отмечает выбор, предлагаемый игроку в конце каждой игры: 1 – игра продолжается, 2 – игра заканчивается, 3 – выводится список 5 лучших игроков.

Переменная *id* проверяет корректность ввода игрока при выборе игрока: если *osch* равна 1 или 2, то ввод корректен. В процедуре *AIMove* присутствует переменная *nv*, в которой хранится номер игрока соперника. Также в этой процедуре используется переменная *k*, которая используется для фиксации выбора хода компьютера.

Структура элемента списка «List» содержит следующие поля: «name» содержит имя игрока, «count» - число его побед, «next» - указатель на следующий элемент списка.

Реализация структуры:

```
struct Node {  
    int count = 0;  
    char player[20];  
    Node* next;  
};
```

Для игры против компьютера был разработан простой искусственный интеллект. Если компьютера ходит первым, то в первую очередь он стремится занять центр или углы. Если игрок сделал ход, компьютера будет блокировать противника.

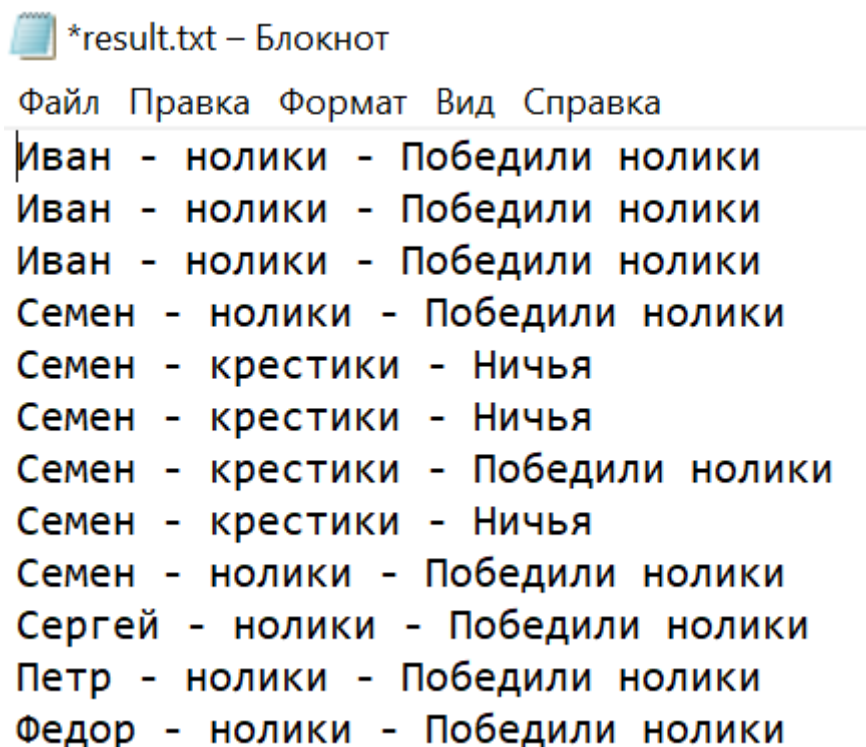


Рисунок 3. Файл с результатами игр

Руководство пользователя

При запуске игры откроется заставка с именем создателя игры и меню с вводом имени (рисунок 3).

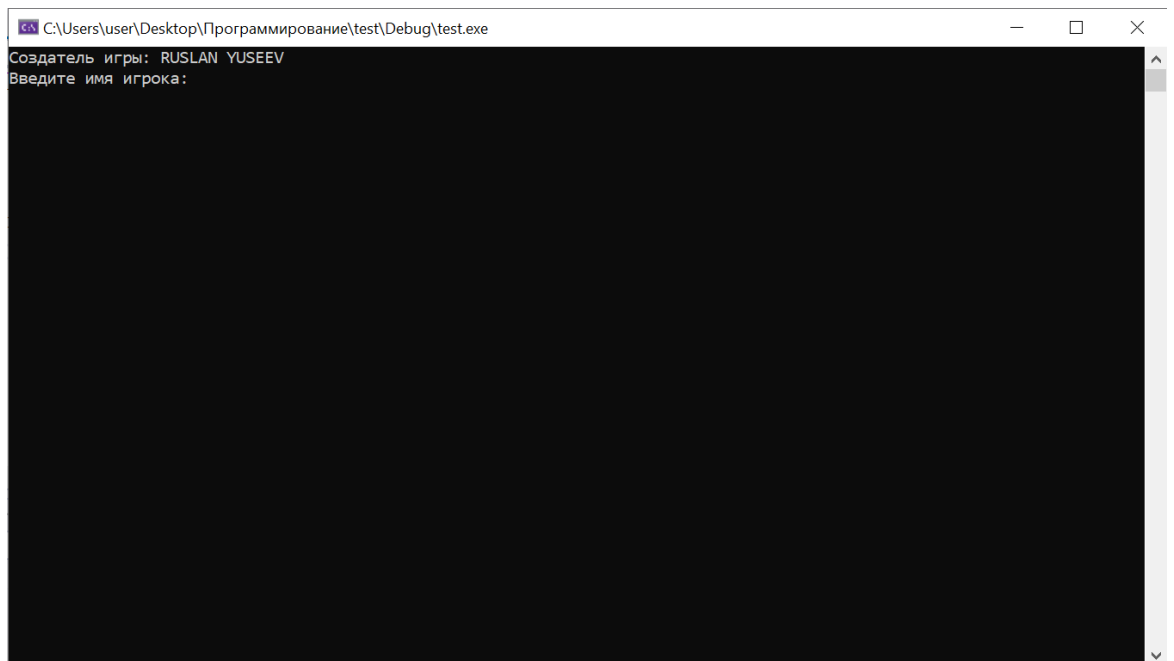


Рисунок 4.

После ввода имени игроку предлагается выбрать фигуру (рисунок 4).

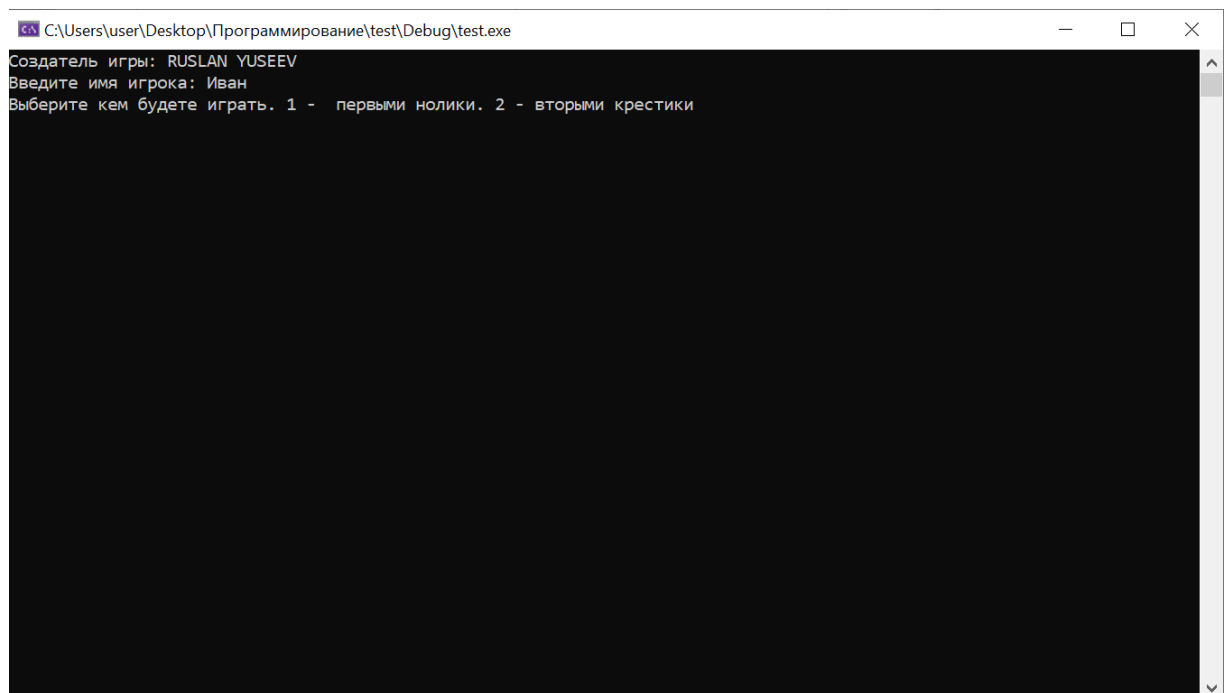


Рисунок 5

После выбора фигуры, выводится игровое поле. Игроку предлагается ввести номер строки и столбца (рисунок 5).

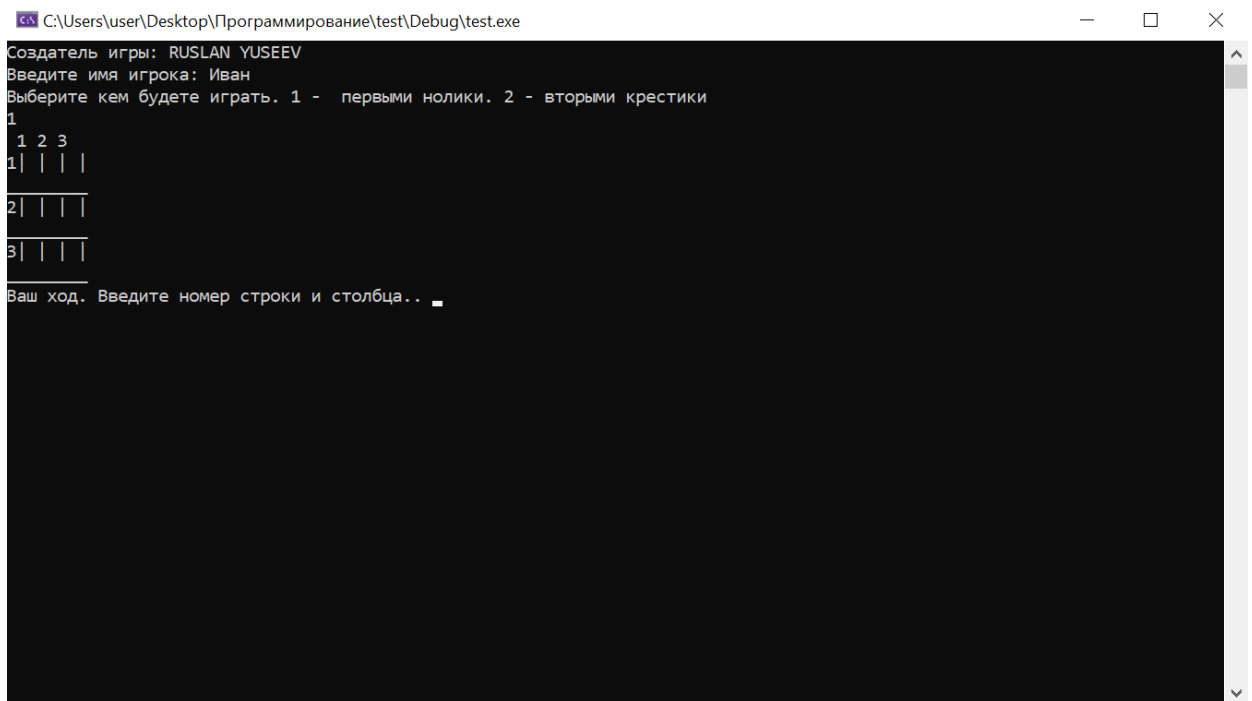


Рисунок 6

Когда игрок ввел номер строки и столбца поля, программа ставит фигуру в то же положение. Сразу же свой ход делает компьютер (рисунок 6).

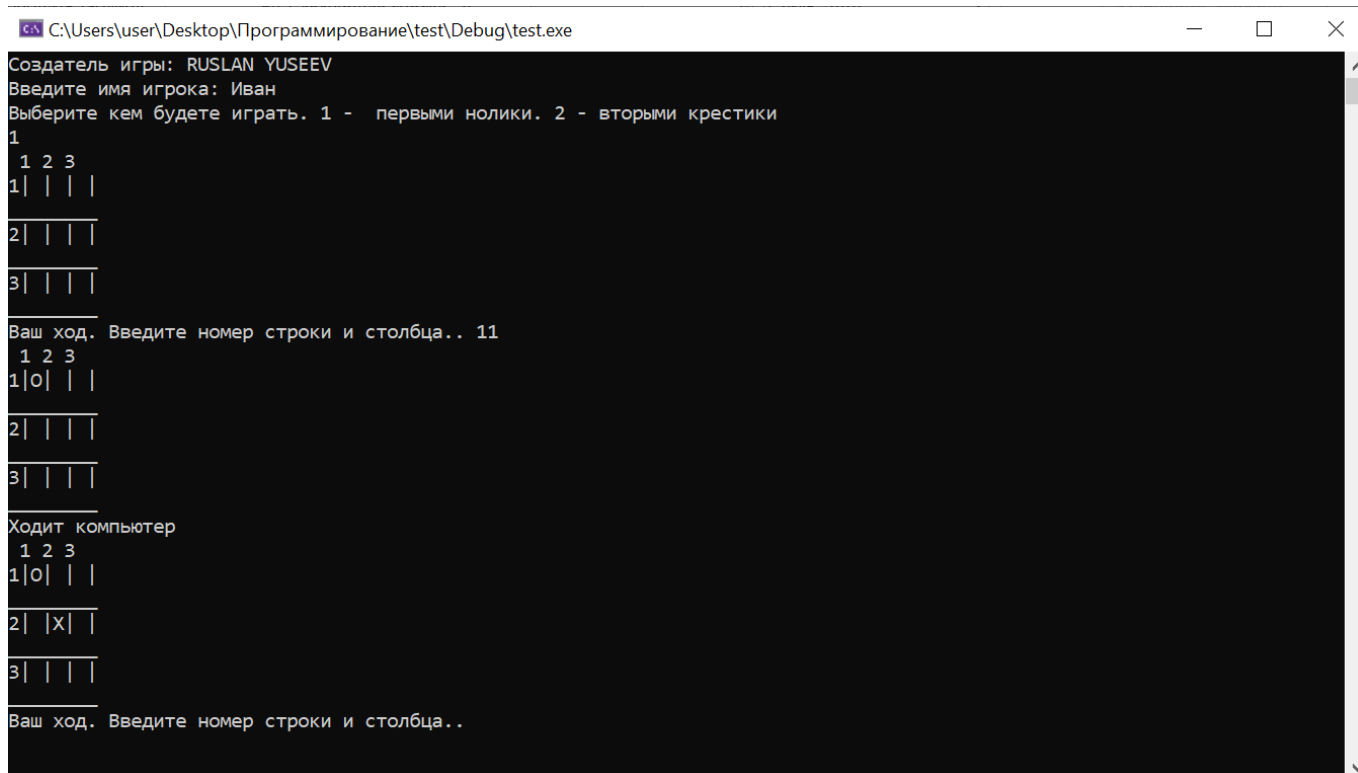
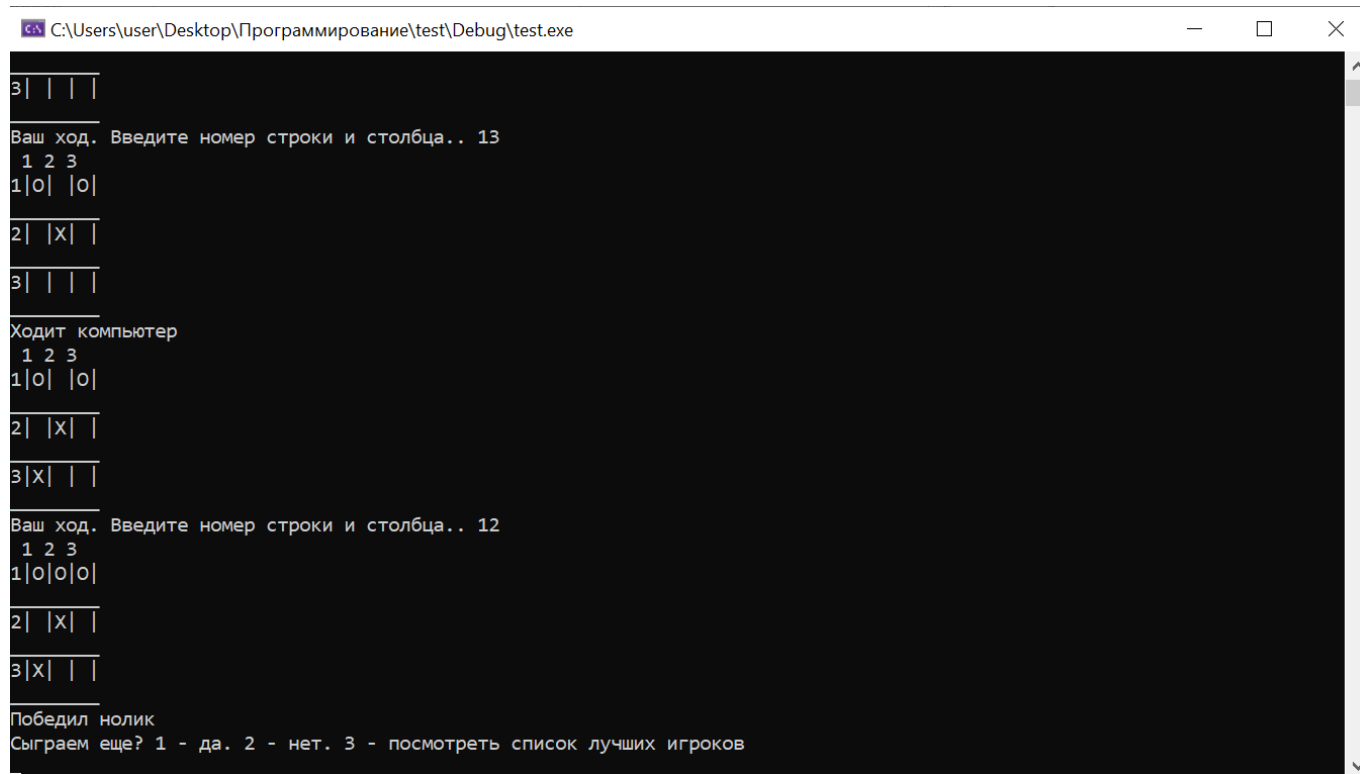


Рисунок 7

Игра продолжается до тех пор, пока один из игроков не выстроит в ряд свои три фигуры, или пока на поле не останется свободного места (рисунок 7).



```
C:\Users\user\Desktop\Программирование\test\Debug\test.exe

3| | |
|_|_|_|
Ваш ход. Введите номер строки и столбца.. 13
 1 2 3
1|O| |O|
|_|_|_|
2| |X| |
|_|_|_|
3| | | |
|_|_|_|
Ходит компьютер
 1 2 3
1|O| |O|
|_|_|_|
2| |X| |
|_|_|_|
3|X| | |
|_|_|_|
Ваш ход. Введите номер строки и столбца.. 12
 1 2 3
1|O|O|O|
|_|_|_|
2| |X| |
|_|_|_|
3|X| | |
|_|_|_|
Победил нолик
Сыграем еще? 1 - да. 2 - нет. 3 - посмотреть список лучших игроков
```

Рисунок 8

После победы одного из игроков или ничьи, игроку выводится меню, в котором предлагается продолжить игру, выйти из игры или вывести список лучших пяти игроков (рисунок 9).

```
C:\Users\user\Desktop\Программирование\test\Debug\test.exe

3|x| |o|
-----
Ходит компьютер
 1 2 3
1|o| |o|
-----
2|o|x|x|
-----
3|x|x|o|
-----
Ваш ход. Введите номер строки и столбца.. 21
Ваш ход. Введите номер строки и столбца.. 12
 1 2 3
1|o|o|o|
-----
2|o|x|x|
-----
3|x|x|o|
-----
Победил нолик
Сыграем еще? 1 - да. 2 - нет. 3 - посмотреть список лучших игроков
3
Иван - 3 побед
Семен - 2 побед
Сергей - 1 побед
Петр - 1 побед
Федор - 1 побед
Сыграем еще? 1 - да. 2 - нет. 3 - посмотреть список лучших игроков
```

Рисунок 9

Разработка функции на языке Assembler

В данной программе ассемблер используется для вывода имени создателя игры.

Пример:

```
char piu[] = "Создатель игры: RUSLAN YUSEEV\n";
char fmt[] = "%s";
_asm
{
    lea edx, piu
    push edx;
    call dword ptr printf
    pop edx
    push eax
    lea ebx, fmt
    push ebx
    add esp, 4
}
```

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Были освоена работа с файлами и массивами, изучены принципы работы структур и списков. Изучены способы обработки событий с клавиатуры. Получены базовые навыки программирования на языке Си и Ассемблера. Также были получены основные навыки отладки и тестирования программ

В рамках курсовой работы была написана программа для игры в крестики-нолики. Данная программа не требует большого количества системных ресурсов, имеет упрощенный интерфейс и управление с помощью клавиатуры.

Список используемых источников

1. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.
2. А.А. Тюгашев. Языки программирования. Учебное пособие. 2018 г.
3. Руслан Аблязов. Программирование на ассемблере на платформе x86-64. 2016 г.

Приложение А. Листинги программы

Файл «TicTacToe.cpp»:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
#include <string.h>
#include <windows.h>
#include <iostream>
//подключение собственных модулей
#include "GameField.h"
#include "PlayerMove.h"
#include "AIMove.h"
#include "CheckStat.h"
#include "Result.h"
#include "TopList.h"
// объявляем текущее поле 3 строки и 3 столбца. если нет ничего то поле 0, если нолик,
то 1, если крестик то 2
int a[3][3] = {};
Node* Head = NULL;
int main()
{
    setlocale(LC_ALL, "Russian");
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    int och = 0; // здесь хранится значение, за кого мы играе
    int v = 0; // если = 1, то играем ещё, если = 2, то прекращаем игру.
    int igra = 0; // состояние игры если победил нолик то 1, если победил крестик то 2
    // если никто пока не победил то 0
    int ud = 0; // проверка корректности ввода
    char name[20];
    char piu[] = "Создатель игры: RUSLAN YUSEEV\n";
    char fmt[] = "%s";
    _asm
    {
        lea edx, piu //в edx записываем строку h
        push edx;
        call dword ptr printf //смотрит на верх стека и выводит последнее, что записали
туда
        pop edx
        push eax
        lea ebx, fmt //формат для ввода
        push ebx
        add esp, 4 //убираем 2 верхних значения из стека
    }
    v = 1;
    while (v == 1)
    {
        ud = 0;
        printf("Введите имя игрока: ");
        scanf("%s", name);
        while (ud == 0)
        {
            printf("Выберите кем будете играть. 1 - первыми нолики. 2 - вторыми
крестики\n");
            scanf("%d", &och);
            if (och == 1 || och == 2) { ud = 1; }
        }
    }
```

```

// обнуляем очищаем поле
nul(a);
// отображаем игровое поле
visual(a);
igra = 0;
// пока никто не победил продолжается игра
while (igra == 0)
{
    if (och == 1) {
        // ход 1 го игрока нолика
        PlayerMove(1, a);
        // отображаем игровое поле
        visual(a);
        // проверка на победу первого игрока
        igra = CheckStat(a);
        // если первый игрок не победил , то ходит второй игрок
        if (igra == 0)
        {
            printf("Ходит компьютер\n");
            // ход второго игрока
            AIMove(2, a, sec);
            // отображаем игровое поле
            visual(a);
            // проверка на победу второго игрока
            igra = CheckStat(a);
        }
    }
    if (och == 2) {
        // ход 1 го игрока нолика
        printf("Ходит компьютер\n");
        AIMove(1, a, sec);
        // отображаем игровое поле
        visual(a);
        // проверка на победу первого игрока
        igra = CheckStat(a);
        // если первый игрок не победил, то ходит второй игрок
        if (igra == 0)
        {
            // ход второго игрока
            PlayerMove(2, a);
            // отображаем игровое поле
            visual(a);
            // проверка на победу второго игрока
            igra = CheckStat(a);
        }
    }
}
// отображение результатов игры
if (igra == 1) { printf("Победил нолик\n"); };
if (igra == 2) { printf("Победил крестик\n"); };
if (igra == 3) { printf("Ничья "); };
if ((igra == 1 && och == 1) || (igra == 2 && och == 2)) FormList(Head, name);
result_to_file(igra, och, name);
printf("Сыграем еще? 1 - да. 2 - нет. 3 - посмотреть список лучших игроков \n");
scanf("%d", &v);
while (v == 3) {
    List(Head);
    printf("Сыграем еще? 1 - да. 2 - нет. 3 - посмотреть список лучших игроков
\n");
    scanf("%d", &v);
}

```

```

    }
    getch();
    return 0;
}

```

Файл «GameField.cpp»:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void visual(int a[][3]) //----- вывод поля на экран
{
    int i, j;
    printf(" 1 2 3\n");
    // пробегаем по всем строкам
    for (i = 0; i < 3; i++)
    {
        printf("%d", i + 1);
        // пробегаем по всем столбцам
        for (j = 0; j < 3; j++)
        {
            if (a[i][j] == 0) { printf("| "); }
            if (a[i][j] == 1) { printf("|O"); }
            if (a[i][j] == 2) { printf("|X"); }
        }
        // переходим на следующую строку
        printf("\n");
        printf("_____\n");
    }
}
void nul(int a[][3]) //----- обнуление поля
{
    int i, j;
    // пробегаем по всем строкам
    for (i = 0; i < 3; i++)
    {
        // пробегаем по всем столбцам
        for (j = 0; j < 3; j++)
        {
            a[i][j] = 0;
        }
    }
}

```

Файл «PlayerMove.cpp»:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void PlayerMove(int n, int a[][3]) //----- Ход игрока n - номер игрока
{
    int hi; // ход игрока
    int sto; // номер столбца в ходе
    int str; // номер строки в ходе
    int ver; // проверка на корректность хода если нельзя такой ход то 0 , если можно то
1
    ver = 0;

```

```

// запрашиваем ход, пока не будет введен корректный ход
while (ver == 0)
{
    // ввод хода
    printf("Ваш ход. Введите номер строки и столбца.. ");
    scanf("%d", &hi);
    // определяем номер столбца - это последняя цифра в ходе
    sto = (hi % 10) - 1;
    // определяем номер строки хода - это первая цифра
    str = ((hi - sto) / 10) - 1;
    // проверка корректности хода, клетка должна быть пуста и номер должен быть один
    из номеров таблицы 3 на 3
    if (a[str][sto] == 0 && (hi == 11 || hi == 12 || hi == 13 || hi == 21 || hi == 22
    || hi == 23 || hi == 31 || hi == 32 || hi == 33)) (ver = 1);
}
// если ходил игрок 1 то поле хода 1
if (n == 1) { a[str][sto] = 1; }
if (n == 2) { a[str][sto] = 2; }
}

```

Файл «AIMove.cpp»:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int sec()
{
    time_t t;
    struct tm* t_m;
    t = time(NULL);
    t_m = localtime(&t);
    return t_m->tm_sec;
}
void AIMove(int n, int a[][3], int sec())//-----Ход компьютера
{
    int i, j, k;
    int nv; // против кого мы играем? Если 1, то = 1, то против ноликов, если = 2, то
    против крестиков
    k = 0; // Флаговая переменная. Если 0, то компьютер ещё не сходил, если 1 - компьютер
    сходил и все последующие проверки в этой процедуре пропускаются.
    if (n == 1) { nv = 2; }
    if (n == 2) { nv = 1; }
    // ищем свой выигрышный ход
    // проверяем все столбцы и выбираем оптимальный ход для компьютера
    for (i = 0; i < 3; i++)
    {
        if (a[0][i] == n && a[1][i] == n && a[2][i] == 0 && k == 0) { a[2][i] = n; k = 1;
    }
        if (a[0][i] == 0 && a[1][i] == n && a[2][i] == n && k == 0) { a[0][i] = n; k = 1;
    }
        if (a[0][i] == n && a[1][i] == 0 && a[2][i] == n && k == 0) { a[1][i] = n; k = 1;
    }
    }
    for (i = 0; i < 3; i++)
    {
        if (a[i][0] == n && a[i][1] == n && a[i][2] == 0 && k == 0) { a[i][2] = n; k = 1;
    }
        if (a[i][0] == 0 && a[i][1] == n && a[i][2] == n && k == 0) { a[i][0] = n; k = 1;
    }
        if (a[i][0] == n && a[i][1] == 0 && a[i][2] == n && k == 0) { a[i][1] = n; k = 1;
    }
    }
}

```

```

// блокируем противника
// проверяем все строки и выбираем оптимальный ход для компьютера
for (i = 1; i <= 3; i++)
{
    if (a[0][i] == nv && a[1][i] == nv && a[2][i] == 0 && k == 0) { a[2][i] = n; k =
1; }
    if (a[0][i] == 0 && a[1][i] == nv && a[2][i] == nv && k == 0) { a[0][i] = n; k =
1; }
    if (a[0][i] == nv && a[1][i] == 0 && a[2][i] == nv && k == 0) { a[1][i] = n; k =
1; }
}
for (i = 1; i <= 3; i++)
{
    if (a[i][0] == nv && a[i][1] == nv && a[i][2] == 0 && k == 0) { a[i][2] = n; k =
1; }
    if (a[i][0] == 0 && a[i][1] == nv && a[i][2] == nv && k == 0) { a[i][0] = n; k =
1; }
    if (a[i][0] == nv && a[i][1] == 0 && a[i][2] == nv && k == 0) { a[i][1] = n; k =
1; }
}
// если свободен центр, занимаем его
if (k == 0 && a[1][1] == 0 && (rand() + sec()) % 7 > 1) { a[1][1] = n; k = 1; };
// если свободно занимаем углы
if (k == 0 && a[0][0] == 0 && (rand() + sec()) % 5 > 1) { a[0][0] = n; k = 1; };
// если свободно занимаем углы
if (k == 0 && a[0][2] == 0 && (rand() + sec()) % 5 > 1) { a[0][2] = n; k = 1; };
// если свободно занимаем углы
if (k == 0 && a[2][0] == 0 && (rand() + sec()) % 5 > 1) { a[2][0] = n; k = 1; };
// если свободно занимаем углы
if (k == 0 && a[2][2] == 0 && (rand() + sec()) % 5 > 1) { a[2][2] = n; k = 1; };
// случайный выбор свободного места
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 3; j++)
    {
        if (a[i][j] == 0 && (rand() + sec()) % 2 == 1 && k == 0)
        {
            a[i][j] = n;
            k = 1;
        }
    }
}
if (k == 0)
{
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            if (a[i][j] == 0 && k == 0)
            {
                a[i][j] = n;
                k = 1;
            }
        }
    }
}
}

```

Файл «CheckStat.cpp»:

```

#include <stdio.h>
#include <conio.h>

```

```

#include <stdlib.h>
int CheckStat(int a[][3]) //----- Проверка. Если ни один игрок не выиграл то
0, если выиграл нолик то 1 если выиграл крестик то 2
{
    int i;
    int res; // значение функции. Если ни один игрок не выиграл то 0, если выиграл нолик
то 1 если выиграл крестик то 2
    res = 0;
    // проверяю выигрыш первого игрока
    // пробегаем по всем строкам
    for (i = 0; i < 3; i++)
    {
        // проверка все ли в строке нолики
        if (a[i][0] == 1 && a[i][1] == 1 && a[i][2] == 1) { res = 1; }
    }
    // пробегаем по всем столбцам
    for (i = 0; i < 3; i++)
    {
        // проверка все ли в столбце нолики
        if (a[0][i] == 1 && a[1][i] == 1 && a[2][i] == 1) { res = 1; }
    }
    // проверка диагоналей
    if (a[0][0] == 1 && a[1][1] == 1 && a[2][2] == 1) { res = 1; }
    if (a[0][2] == 1 && a[1][1] == 1 && a[2][0] == 1) { res = 1; }

    // проверим выигрыш второго игрока
    // пробегаем по всем строкам
    for (i = 0; i < 3; i++)
    {
        // проверка все ли в строке крестики
        if (a[i][0] == 2 && a[i][1] == 2 && a[i][2] == 2) { res = 2; }
    }
    // пробегаем по всем столбцам
    for (i = 0; i < 3; i++)
    {
        // проверка все ли в столбце крестики
        if (a[0][i] == 2 && a[1][i] == 2 && a[2][i] == 2) { res = 2; }
    }
    // проверка диагоналей
    if (a[0][0] == 2 && a[1][1] == 2 && a[2][2] == 2) { res = 2; }
    if (a[0][2] == 2 && a[1][1] == 2 && a[2][0] == 2) { res = 2; }
    // условие ничьи
    if (a[0][0] * a[0][1] * a[0][2] * a[1][0] * a[1][1] * a[1][2] * a[2][0] * a[2][1] *
a[2][2] > 0 && res == 0) { res = 3; }
    return res;
}

```

Файл «Result.cpp»:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
void result_to_file(int igra, int och, char name[])
{
    FILE* fin = fopen("result.txt", "a");
    char player[20];
    char stat[20];
    if (och == 1) {
        strcpy(player, "нолики");
    }
    else strcpy(player, "крестики");
    if (igra == 1) strcpy(stat, "Победили нолики");
}

```

```

else if (igra == 2) strcpy(stat, "Победили крестики");
else strcpy(stat, "Ничья");
fprintf(fin, "%s - %s - %s\n", name, player, stat);
fclose(fin);
}

```

Файл «TopList.cpp»:

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
struct Node {
    int count = 0;
    char player[20];
    Node* next;
};
Node* CreateNode(char player[], int count) { //создание узла списка
    Node* NewNode = (Node*)malloc(sizeof(Node));
    strcpy(NewNode->player, player);
    NewNode->count = count;
    NewNode->next = NULL;
    return NewNode;
}
void AddNode(Node* &Head, Node* NewNode) { //добавление узла в список
    Node* q = Head;
    if (Head == NULL) {
        NewNode->next = Head;
        Head = NewNode;
        return;
    }
    while (q->next != NULL) q = q->next;
    NewNode->next = q->next;
    q->next = NewNode;
}
Node* SortNode(Node* &Head) { // сортировка списка
    Node* p = Head;
    Node* temp = Head;
    Node* p1 = p;
    Node* q = NULL;
    bool check;
    char name[20];
    int max = 0;
    while(temp != NULL) {
        max = 0;
        p1 = p;
        while (p1 != NULL) {
            if (max < p1->count) {
                max = p1->count;
                strcpy(name, p1->player);
            }
            p1 = p1->next;
        }
        Node *NewNode = CreateNode(name, max);
        AddNode(q, NewNode);
        p1 = p;
        while (p1 != NULL) {
            if (p1->count == max && strcmp(p1->player, name) == 0) {
                p1->count = -1;
            }
            p1 = p1->next;
        }
        temp = temp->next;
    }
}

```

```

    }
    return q;
}

void FormList(Node* &Head, char name[]) { // формирование списка
    Node* p = Head;
    Node* p1 = p;
    bool check = true;
    while (p1 != NULL) {
        if (strcmp(p1->player, name) == 0) {
            p1->count++;
            check = false;
            break;
        }
        p1 = p1->next;
    }
    if (check) {
        Node* NewNode = CreateNode(name, 1);
        AddNode(p, NewNode);
    }
    Head = SortNode(p);
}

void List(Node* Head) // списка на экран
{
    Node* p = Head;
    int counter = 0;
    while (p != NULL && counter < 5) {
        printf("%s - %d побед\n", p->player, p->count);
        counter++;
        p = p->next;
    }
}

```

Приложение В. Результат работы программы

Файл result.txt

Иван - нолики - Победили нолики
 Иван - нолики - Победили нолики
 Иван - нолики - Победили нолики
 Семен - нолики - Победили нолики
 Семен - крестики - Ничья
 Семен - крестики - Ничья
 Семен - крестики - Победили нолики
 Семен - крестики - Ничья
 Семен - нолики - Победили нолики
 Сергей - нолики - Победили нолики
 Петр - нолики - Победили нолики
 Федор - нолики - Победили нолики