

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»

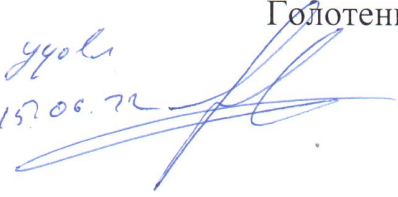
**Выполнил студент группы 21ВВ1:**

Гришин Сергей

**Принял:**

Голотенков Н. О

усл.  
15.06.22



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 Тришину Сергею Вадимовичу Группа 21ВВ1

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

Исходные данные (технические требования) на проектирование

*Приложение: Игровая программа "Змейка".  
Требования: Графический интерфейс, не менее 15 клеток по  
верт. и гор., функция ввода имени, запись результатов  
в файле, заставка при загрузке*

**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 “22” февраля

Дата защиты проекта “15” 06. 22

Руководитель Голоженков Н.О.

Задание получил “22” февраля 2022г.

Студент Гришин С.В. И

## Описание структуры программы

Файл «zmeika» — главный файл. В нём объявляются основные переменные и массивы программы. Функция «main» — главная функция, в которой вызываются функции, определяющие игровой процесс. Структура программы показана на рисунке 1.

Рисунок 1 - Структура программы

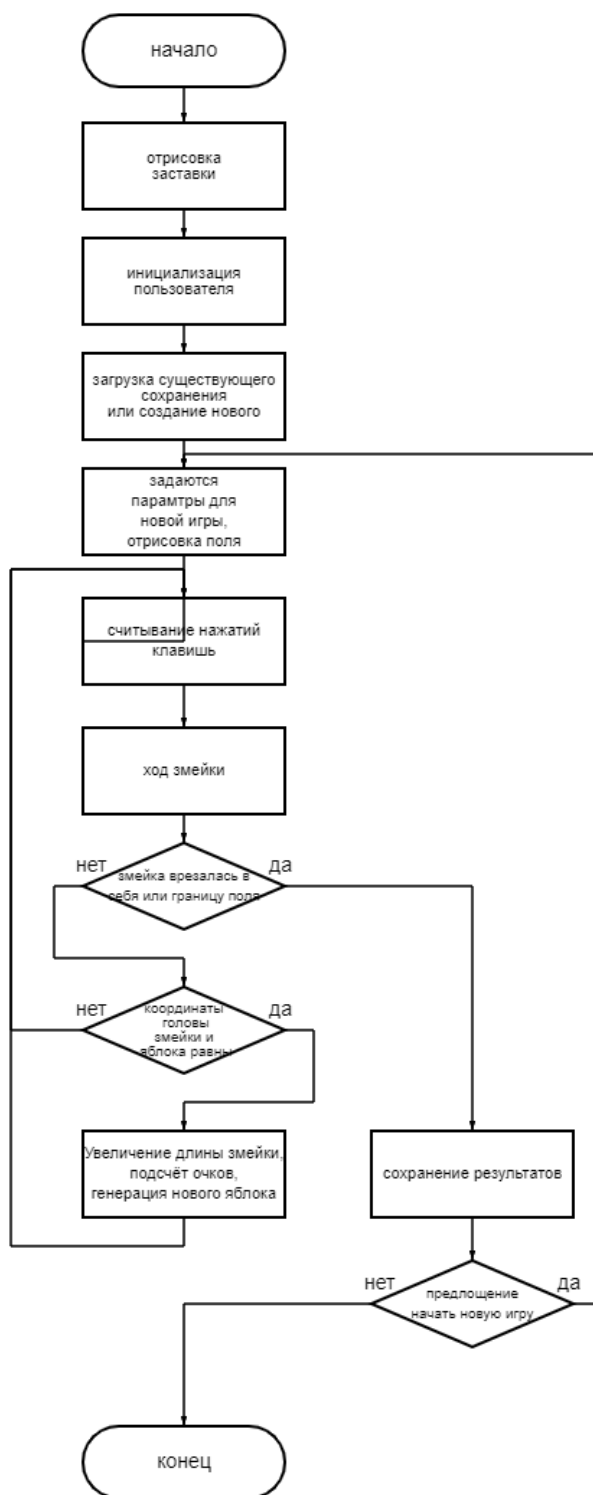


Рисунок 1 – схема программы

Заголовочный файл «draw.h» — правильной отрисовки поля. Включает в себя функции: *clear* используется для удаления предыдущего кадра вместо *system("cls")*, при этом удаляет эффект мерцания поле, делая игру плавной и приятной для глаза; *hidecursore* делает курсор невидимым, убирает эффект “снега”.

Файл «draw.cpp» содержит определение функций, объявленных в «draw.h».

## Описание структуры программы

Функция «main» – главная функция. В ней реализуется весь игровой процесс.

Сначала программа проверяет наличие файла с сохранениями, если файла нет, то он создаётся, а после ввода игроком имени записывается первое “нулевое” сохранение. Если файл с сохранениями присутствует, игра просит ввести игрока имя. После ввода идёт проверка сохранений, если сохранение с таким именем есть, оно загружается, а если такого нет, то создаётся “нулевое сохранение”. Далее вызываются функции *clearp()* и *menuprint()* для очистки консоли от диалога с пользователем.

Выставляются начальные настройки (длина змейки, координаты первого яблока, координаты головы змейки, очки) и запускается цикл с самой игрой.

Сперва вызывается *Sleep()* чтобы создать задержку при выполнении цикла, с помощью которой можно контролировать скорость змейки, частоту обновления консоли и улучшить отзывчивость управления. После вызывается *clear()* для очистки предыдущего кадра. Затем с помощью *creategame()*, *aplgen()* и *createsnake* идёт инициализация игрового поля, яблока и змейки соответственно. Следующая функция *printgame()* отрисовывает кадр. Потом с помощью команды *movesnakehead()* змейка двигается (направление движения зависит от переменного *parp*), а с помощью *eatcheck* идёт проверка “съела ли змейка яблоко”. В конце цикла определяется какая клавиша нажата и выставления определённого значения для переменной *parp* и цикл повторяется.

Цикл с игрой продолжится пока переменная *gameover* имеет значение *false*. Значение это переменной меняется на *false* в двух случаях:

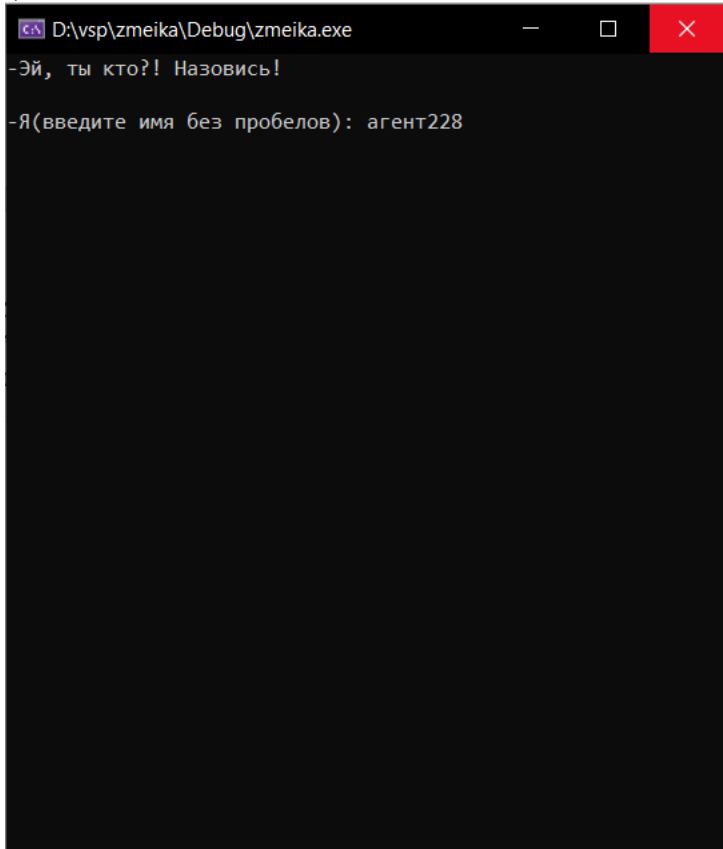
1. Была нажата клавиша “Y”
2. Координаты головы змейки совпадают с координатами границ поля или одной из частей её хвоста.

После завершения цикла идёт сохранение рекорда, если он был поставлен. Так же игроку предлагается начать новую игру. Если игрок соглашается, то задаются параметры для новой игры и начинается цикл с игрой. Если же игрок отказался от предложения начать новую игру, то программа закрывается.

# Руководство пользователя

Игра “Змейка” - 2D игра с элементами пользовательского интерфейса, исполненного в виде диалога между игроком и программой. В этих диалогах присутствует анимация печати текста, которую можно пропустить нажатием клавиши “Пробел”.

После запуска программа попросит вас имя, его нужно вводить без пробелов (максимальная длина имени 25 символов, хватит на любое имя).



Очень редко, перед начало игры или после поражения, может появиться ошибка. Чтобы её решить, на этапе ввода имени на 3-4 секунды зажмите клавишу “backspace”.

## Управление:

Управление осуществляется клавишами W A S D (вверх, влево, вниз и вправо соответственно), также можно прервать игру нажатием клавиши Y.

## Правила:

Управляя змейкой ешьте яблоки(Q), которые появляются на всём игровом поле. Чем больше яблок вы съедите, тем длиннее будет ваша змейка и больше ваш счёт.

Вы проиграете если змейка врежется в одну из стен, которые стоят по краям поля(#) или в собственное тело(O).

После поражения вам предложат ещё одну попытку. Нажмите Y чтобы согласиться или N чтобы отказаться

## тестирование

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

- 1) Тестирование на загрузку и запись сохранения. При запуске программа должна успешно считывает сохранения(имя и рекорд) и записывать полученные в соответствующие переменные, так же программа должна создавать сохранения при необходимости. После завершения игры, программа должна изменять сохранение игрока, записав поставленный рекорд.
- 2) Тестирование на отрисовку. Программа должна правильно отрисовывать объекты. Сначала рисуется поле, потом яблоко, и в конце сама змейка, а в верхней части окна отображается информация о имени, набранных очках и рекорде игрока
- 3) Тестирование на правильное движение змейки и управление. Змейка должна максимально быстро реагировать на нажатие клавиш, при это двигаться строго в указанном направлении. При это хвост змейки следует за ней и сохраняет свою длину.
- 4) Тестирование на спавн яблок. Яблоко должно спавниться только на свободной ячейке поля. При этом на поле всегда должно быть одно яблоко.
- 5) Тестирование на проигрыш. Игра должна останавливаться при столкновении змейки со стеной или своим телом.

| Описание теста               | Ожидаемый результат                                                           | Полученный результат |
|------------------------------|-------------------------------------------------------------------------------|----------------------|
| Загрузка и запись сохранения | Корректная загрузка и запись сохранения                                       | верно                |
| Отрисовка поля               | Корректно отрисовывается поле, яблоко и змейка                                | верно                |
| Движение змейки и управление | Змейка имеет правильную механику движения и реагирует на нажатие клавиш       | верно                |
| Спавн яблок                  | Корректно спавняются яблоки                                                   | верно                |
| Конец игры                   | Игра должна останавливаться при столкновении змейки со стеной или своим телом | верно                |

## Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Изучены способы обработки событий с клавиатуры. Получены знания и навыки по работе со структурами. Получены базовые навыки программирования на языке Си и Ассемблера. Получены навыки отладки и тестирования программ. В рамках данной курсовой была написана игра “Змейка”. Данная программа не требует большого количества системных ресурсов, имеет символьную графику и управление с помощью клавиш. Программа сделана с учётом на возможность добавления новых возможностей: таблица лидеров, настройка начальных параметров, изменение графики, редактор уровней, совместная игра с одного устройства.

## Листинг программы

### Файл “zmeika”:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <conio.h>
#include <Windows.h>
#include <time.h>
#include <malloc.h>
#include "draw.h"

#define h 30
#define w 60

//скины\
char sapl = 'Q'; //скин яблока
char ssnakehead = '@'; //скин головы змеи
char ssnaketail = 'O'; //скин хвоста змеи
char sgr = '#'; //скин границ карты

int score = 0, len = 0, i = 0, j = 0, akkload = 0, oldrec = 0, n = 0, nachlen = 5, textlen = 0, skip = 20;
char pole[h][w];
short int napr = 0; //направление
short int crtime = 0; //время движения
bool gameover = false;
char text[60];

//змейка\
typedef struct snst {
    int x;
    int y;
    char jb = ssnaketail;
} snst;
//яблоко\
typedef struct aps {
    short int x;
    short int y;
    bool gen;
    char jb = sapl;
} apss;
//Аккаунты\
typedef struct aks {
    char name[25];
    char namereccheck[46];
    char namecheck[31];
    char reccheck[14];
    int record;
} ak;

ak akk, *akks;
apss apl;
snst snake[1600];
snst bfs;
```

```

//создание границ поля\
void creategame()
{
    for (int i = 0; i < h-2; i++)
    {
        for (int j = 0; j < w; j++)
        {
            pole[i][j] = sgr;
        }
    }

    for (int i = 1; i < h-3; i++)
    {
        for (int j = 1; j < w-1; j++)
        {
            pole[i][j] = ' ';
        }
    }
}

void clearp()
{
    for (int i = 0; i < h; i++)
    {
        for (int j = 0; j < w; j++)
        {
            pole[i][j] = ' ';
        }
    }
}

//отрисовка поля\
void printprgame()
{
    printf_s("тебя зовут %s\n", akk.name);
    printf("ты набрал %d, а твой рекорд %d \n", score, akk.record);
    for (int i = 0; i < h-2; i++)
    {
        for (int j = 0; j < w; j++)
        {
            printf("%c", pole[i][j]);
        }
        if (i != h-3) { printf("\n"); }
    }
}

void printpmenu()
{
    for (int i = 0; i < h; i++)
    {
        for (int j = 0; j < w; j++)
        {
            printf("%c", pole[i][j]);
        }
        if (i != h-1) { printf("\n"); }
    }
}

//отрисовка змейки\
void createsnake()
{
    for (int i = 0; i <= len; i++)
    {
        if ((snake[i].x != 0) and (snake[i].y != 0))
        {
            pole[snake[i].x][snake[i].y] = snake[i].jb;
        }
    }
}

//движение змейки\
void movesnakehead()
{
    crtime++;
    if (crtime == 2)
    {
        if (napr == 1) {
            if ((pole[snake[0].x - 1][snake[0].y] != sgr) and (pole[snake[0].x - 1][snake[0].y] != ssnaketail))
            {
                bfs.x = snake[0].x;
                bfs.y = snake[0].y;
                snake[0].x = snake[0].x - 1;
            }
            else { gameover = true; }
        }
        if (napr == 2)
        {
            if ((pole[snake[0].x + 1][snake[0].y] != sgr) and (pole[snake[0].x + 1][snake[0].y] != ssnaketail))

```

```

        {
            bfs.x = snake[0].x;
            bfs.y = snake[0].y;
            snake[0].x = snake[0].x + 1;
        }
        else { gameover = true; }
    }
    if (napr == 3)
    {
        if ((pole[snake[0].x][snake[0].y-1] != sgr) and (pole[snake[0].x][snake[0].y-1] != ssnaketail))
        {
            bfs.x = snake[0].x;
            bfs.y = snake[0].y;
            snake[0].y = snake[0].y - 1;
        }
        else { gameover = true; }
    }
    if (napr == 4)
    {
        if ((pole[snake[0].x][snake[0].y+1] != sgr) and (pole[snake[0].x][snake[0].y+1] != ssnaketail))
        {
            bfs.x = snake[0].x;
            bfs.y = snake[0].y;
            snake[0].y = snake[0].y + 1;
        }
        else { gameover = true; }
    }
    crtime = 0;
}

//генерация яблок\
void aplgen()
{
    if (apl.gen == false)
    {
        apl.x = 1+rand() % (h-4);
        apl.y = 1+rand() % (w-2);
        pole[apl.x][apl.y] = apl.jb;
        apl.gen = true;
        for (int n = 0; n < len; n++)
        {
            if ((apl.x == snake[n].x) and (apl.y == snake[n].y)) { apl.gen = false; }
        }
    }
    else
    {
        pole[apl.x][apl.y] = apl.jb;
    }
}

//змея жрёт тут\
void eatcheck()
{
    if ((snake[0].x == apl.x) and (snake[0].y == apl.y))
    {
        apl.gen = false;
        score++;
        if (score > akk.record)
        {
            akk.record = score;
            if (oldrec == akk.record)
            {
                oldrec = score;
            }
        }
        len++;
    }
}

//движение хвоста\
void telomove()
{
    if (crtime == 1)
    {
        for (int i = len; i >= 1; i--)
        {
            if (i > 1)
            {
                snake[i].x = snake[i - 1].x;
                snake[i].y = snake[i - 1].y;
            }
            if (i == 1)
            {
                snake[i].x = bfs.x;
                snake[i].y = bfs.y;
            }
        }
    }
}

```

```

    }
}
//анимация текста\
void textanim()
{
    textlen = strlen(text);
    for (int n = 0; n < textlen; n++)
    {
        printf("%c", text[n]);
        if (napr != 2) { if (GetKeyState(VK_SPACE) < 0)(skip=0); }
        Sleep(skip);
    }
    skip = 20;
}

```

```

int main()
{
    FILE* file;
    system("chcp 1251");
    srand(time(NULL));
    system("mode con cols=60 lines=30");
    hidecursor();
    //logo н
    clear(15, 12);
    strcpy(text, "Эта игра была написана учеником\n");
    textanim();
    clear(16, 13);
    strcpy(text, "группы 21ВВ1 Гришиным Сергеем\n");
    textanim();
    clear(24, 14);
    strcpy(text, "в 2022 году\n");
    textanim();
    Sleep(2000);
    system("cls");
    Sleep(400);
    //logo к
    //Загрузка н
    if ((file = fopen("D:/SaveSnake.txt", "r")) == NULL)
    {
        strcpy(text, "-Ты у нас новенький?\n\n");
        textanim();
        Sleep(500);
        strcpy(text, "-Небойся, щас тебя запишем\n\n");
        textanim();
        Sleep(500);
        strcpy(text, "-Как тебя зовут?\n\n");
        textanim();
        Sleep(500);
        strcpy(text, "-Я(введите имя без пробелов: ");
        textanim();
        scanf_s("%s", &akk.name, 25);
        file = fopen("D:/SaveSnake.txt", "w");
        fprintf(file, "%s 0\n", akk.name, 25);
        fclose(file);
        strcpy(text, "\n-Готово, ");
        textanim();
        strcpy(text, akk.name);
        textanim();
        strcpy(text, ", ты в списке, развлекайся\n");
        textanim();
        Sleep(500);
    }
    else
    {
        strcpy(text, "-Эй, ты кто?! Назовись!\n\n");
        textanim();
        strcpy(akk.name, "");
        strcpy(text, "-Я(введите имя без пробелов: ");
        textanim();
        scanf_s("%s", akk.name, 25);
        for (int n = 0; n < 25; n++)
        {
            if (akk.name[n] == '\0') { akkload = 1; }
            if ((akk.name[n] == 'ю') and (akkload == 1))
            {
                akk.name[n] = '\0';
            }
        }
        akkload = 0;
        Sleep(1000);
    }
}

```

```

strcpy(text, "\n-Погоди, я проверю список\n\n");
textanim();
file = fopen("D:/SaveSnake.txt", "r");
while (akkload==0)
{
    strcpy(akk.namereccheck, "");
    fgets(akk.namereccheck, 46, file);
    i = 0;
    if ((akk.namereccheck[0] != '\0') and (akkload != 1))
    {
        while (akk.namereccheck[i] != ' ')
        {
            akk.namecheck[i] = akk.namereccheck[i];
            i++;
        }
    }
    if (strcmp(akk.namecheck, akk.name) == 0)
    {
        akkload = 1;
        i++;
        j = 0;
        while (akk.namereccheck[i] != '\0')
        {
            akk.reccheck[j] = akk.namereccheck[i];
            i++;
            j++;
        }
        akk.record = atoi(akk.reccheck);
        oldrec = akk.record;
        printf("-");
        Sleep(20);
        strcpy(text, akk.name);
        textanim();
        strcpy(text, "... Да, ты в списке проходи");
        textanim();
    }
    i = i;
    if ((akk.namereccheck[0] == '\0') and (akkload!=1))
    {
        akkload = 2;
    }
    for (int n = 0; n < 31; n++)
    {
        akk.namecheck[n] = '\0';
        akk.namereccheck[n] = '\0';
    }
}
fclose(file);
if(akkload==2)
{
    file = fopen("D:/SaveSnake.txt", "a+");
    fprintf(file, "%s 0\n", akk.name, 14);
    fclose(file);
    printf("-");
    Sleep(20);
    strcpy(text, akk.name);
    textanim();
    strcpy(text, "... Тебя нет в списке... Сейчас запишем\n");
    textanim();
}
Sleep(1000);
}
//Загрузка к
res:
system("cls");
clearp();
printpmenu();
clear(0, 0);
gameover = false;
//настройки игры n
len = nachlen;
apl.gen = false;
snake[0].jb = ssnakehead;
/////////////////////////////////АСЕМБЛЕР H
_asm {
    mov eax, h
    mov ebx, 2
    sub eax, ebx
    cdq
    idiv ebx
    mov snake[0].x, eax

```

```

        mov eax, w
        cdq
        idiv ebx
        mov snake[0].y, eax
    }
    ///////////////////////////////////АСЕМБЛЕР К
    if (len != 0)
    {
        for (int i = 1; i <= len; i++)
        {
            snake[i].x = snake[i - 1].x + 1;
            snake[i].y = snake[i - 1].y;
        }
    }
    creategame();
    //настройки игры к
    ///////////////////////////////////основная игра начало
    while (gameover == false)/////////////////////////////////основная игра начало
    {
        Sleep(0.1);
        clear(0,0);
        creategame();
        aplgen();
        createsnake();
        printpgame();
        movesnakehead();
        if (napr != 0) { telomove(); }
        eatcheck();
        if (GetKeyState('Y') < 0)(gameover = true);
        if (napr != 2) { if (GetKeyState('W') < 0)(napr = 1); }
        if ((napr != 1) and (napr != 0)) { if (GetKeyState('S') < 0)(napr = 2); }
        if (napr != 4) { if (GetKeyState('A') < 0)(napr = 3); }
        if (napr != 3) { if (GetKeyState('D') < 0)(napr = 4); }
    }/////////////////////////////////основная игра конец
    ///////////////////////////////////сохранение начало
    if (score > oldrec)
    {
        i = 0;
        j = 0;
        file = fopen("D:/SaveSnake.txt", "a+");
        fgets(akk.namereccheck, 20, file);
        while (akk.namereccheck[0] != '\0')
        {
            strcpy(akk.namereccheck, "");
            fgets(akk.namereccheck, 20, file);
            i++;
        }
        fclose(file);
        file = fopen("D:/SaveSnake.txt", "r+");
        akks = (ak*)malloc(i * sizeof(ak));
        for (j = 0; j < i; j++)
        {
            for (int n = 0; n < 34; n++)
            {
                akks[j].namereccheck[n] = '\0';
                akks[j].namecheck[n] = '\0';
            }
            fgets(akks[j].namereccheck, 34, file);
            n = 0;
            while (akks[j].namereccheck[n] != ' ')
            {
                akks[j].namecheck[n] = akks[j].namereccheck[n];
                n++;
            }
        }
        fclose(file);
        file = fopen("D:/SaveSnake.txt", "w");
        for (j = 0; j < i; j++)
        {
            if (strcmp(akks[j].namecheck, akk.name) != 0)
            {
                fprintf(file, "%s", akks[j].namereccheck, 34);
            }
            else
            {
                fprintf(file, "%s %d\n", akks[j].namecheck, akk.record, 34);
            }
        }
        fclose(file);
        ///////////////////////////////////сохранение конец
    }
    system("cls");
    Sleep(500);

```

```

strcpy(text, "-Ммм... ты проиграл...\n\n");
textanim();
Sleep(500);
if(score>oldrec)
{
    strcpy(text, "-Но у тебя новый рекорд! Поздравляю!");
    textanim();
    strcpy(text, "\n\n-Может попробуешь ещё раз?\n");
    textanim();
    oldrec = score;
}
else
{

    strcpy(text, "-И рекорд новый не поставил...");
    textanim();
    Sleep(500);
    strcpy(text, "\n\n-Может попробуешь ещё раз?\n");
    textanim();
}
Sleep(500);
printf("\nY-да N-нет\n");
i = 0;
while(i!=1)
{
    if (GetKeyState('Y') < 0)
    {
        gameover = false;
        napr = 0;
        for (int n = 0; n < len; n++)
        {
            snake[n].x = 0;
            snake[n].y = 0;
        }
        score = 0;
        goto res;
        i = 1;
    }
    if(GetKeyState('N') < 0)
    {
        strcpy(text, "\n\n-ок,удачи");
        textanim();
        Sleep(1000);
        i = 1;
    }
}
}

```

## Файл “draw.h”

```

#pragma once
void hidecursor();
void clear(int x, int y);

```

## Файл “draw.cpp”

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <conio.h>
#include <Windows.h>
#include <time.h>
#include <malloc.h>
#include "draw.h"

void hidecursor()
{
    HANDLE consoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_CURSOR_INFO info;
    info.dwSize = 100;
    info.bVisible = FALSE;
    SetConsoleCursorInfo(consoleHandle, &info);
}

void clear(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

```