

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему: Игра «Змейка»

Выполнил:

020724
[Signature]

студент группы 23BVB2

Кандаков А.А.

Принял:

к/н., доцент Слепцов Н.В.

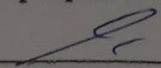
Пенза 2024

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 Хамраеву Андрею Александровичу Группа 23ВВВ2

Тема проекта: Игра «Змейка».

Исходные данные (технические требования) на проектирование

Программа должна предусматривать наст. уровень сложности с различными размерами поля и скоростью передвижения, максимальным количеством размещаемых объектов и вращением игры, предусмотреть запись лучших результатов и цветовое оформление.

Обязательные требования к программе:

1. Управление движением змейки осуществляется с помощью клавиатуры.
2. Система набора очков.
3. Должен быть предусмотрен рост и уменьшение размеров змейки.
4. Изменение цветовой палитры

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 “ ”

Дата защиты проекта “ ”

Руководитель _____

Задание получил “ ” 20__ г.

Студент Александров Андрей Александрович

Содержание

Введение.....	4
1 Постановка задачи.....	7
2 Выбор решения.....	8
2.1 Определение необходимых модулей программы.....	8
3 Описание разработки программы.....	10
4 Отладка и тестирование.....	12
5 Описание программы.....	13
6 Руководство пользователя.....	14
Заключение.....	19
Список используемых источников.....	20
Приложение А.....	21
Приложение В.....	41
Схемы работы программы.....	43

Введение

Данная курсовая работа ставит перед собой две ключевые цели, направленные на приобретение практических навыков и углубленное понимание основ разработки игр.

Цель 1: Создание игры Змейка

Создание игры Змейка является первой и основной целью данной курсовой работы. Эта классическая игра, одна из первых в истории игровой индустрии, является отличным примером простоты и увлекательности игрового процесса. Разработка данной игры предоставляет уникальную возможность изучить и освоить ключевые аспекты разработки игр.

Простота и увлекательность: Основной принцип игры заключается в управлении змейкой, которая движется по экрану, поедая еду и избегая столкновений с собственным хвостом. С каждым съеденным объектом змейка становится длиннее, что увеличивает сложность игры. Эта простая, но увлекательная механика позволяет игрокам сосредоточиться на навыках и стратегии, что делает игру захватывающей и востребованной на протяжении десятилетий.

Практические навыки: Разработка игры Змейка включает в себя следующие ключевые аспекты:

Проектирование игрового процесса: Создание логики игры, включающей правила движения змейки, генерирование объектов на игровом поле и управление очками игрока.

Работа с графикой: Разработка визуального представления змейки, объектов и игрового поля с использованием графических библиотек.

Управление вводом: Реализация механизмов управления змейкой с

помощью клавиатуры или сенсорного экрана, что требует знания обработки пользовательских событий.

Обработка коллизий: Разработка логики для обнаружения столкновений змейки с собственным хвостом, что является ключевым элементом игрового процесса.

Цель 2: Приобретение практических навыков работы с основными игровыми механизмами

Помимо создания самой игры, данная курсовая работа нацелена на глубокое изучение и понимание основных механизмов.

Обработка ошибок: Разработка механизмов обработки ошибок и исключений для обеспечения стабильности и надежности работы игры.

Тестирование и отладка: Применение методов тестирования и отладки для выявления и исправления ошибок в коде, что является важным этапом в процессе разработки.

Технологии и инструменты: в рамках данной курсовой работы будут изучены и использованы современные технологии и инструменты разработки игр, включая языки программирования, графические и физические движки, среды разработки и системы управления версиями. Это позволит приобрести практические навыки работы с профессиональными инструментами, используемыми в индустрии.

Результат работы: Итогом данной курсовой работы станет полнофункциональная игра Змейка, реализованная с использованием современных технологий и подходов к разработке игр. Полученные знания и навыки смогут быть применены в дальнейших проектах и профессиональной деятельности, а также послужат основой для углубленного изучения разработки игр.

1 Постановка задачи

Необходимо разработать игру «Змейка». В игре будут реализованы классические правила:

- «змейка» при приближении к границе игрового поля, не будет врезаться в него, а будет выходить с другой стороны поля;
- основной целью игры является увеличение змейки до, как можно большего размера, путем сбора фруктов;
- за каждый собранный фрукт пользователю будут начисляться очки.
- сам фрукт должен генерироваться в случайном месте;
- «змейка» не может проходить через саму себя, поэтому при столкновении со своим туловищем будет наступать конец игры;
- должна быть создана таблица лидеров;
- должны выводиться очки;

Необходимо также реализовать заставку для приложения, с помощью которой можно будет запустить игру или узнать необходимую информацию.

Программа должна быть многомодульной. Это облегчит обнаружение и исправление ошибок при её отладке и тестировании. Также это повысит расширяемость программы и локализации ошибок при отладке и тестировании программы.

В программе будут использоваться сложные типы данных.

Устройством управления игрой является клавиатура. Все действия в игре должны осуществляться с помощью нажатия клавиш клавиатуры.

2 Выбор решения

2.1 Определение необходимых модулей программы

Разработанная программа состоит из следующих модулей:

- Header.h;
- Function1.h;
- Source.cpp;
- Function1.cpp.

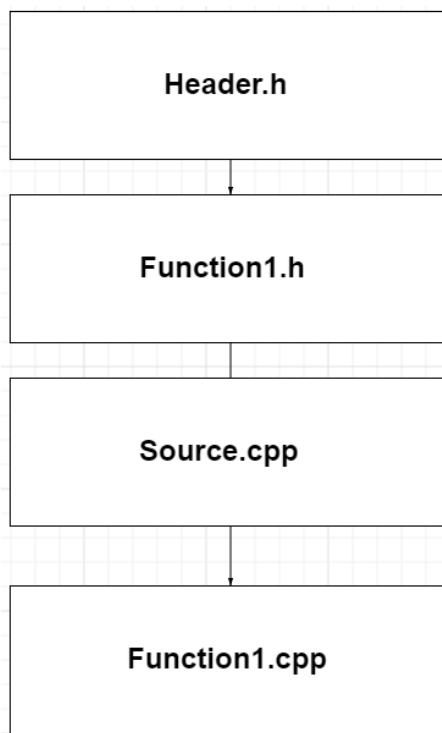


Рисунок 1 – Схема взаимодействия системы

Header.h – в этом модуле располагаются библиотеки, требуемые для функционирования программы.

Function1.cpp – в данном модуле добавляются X и Y для вывода каких-либо данных в конкретное место в командной строке.

Source.cpp – является основным модулем программы. Именно здесь происходит генерация процесса игры, с непосредственным использованием модулей Header.h и Function1.cpp.

3 Описание разработки программы

Для написания программы были использованы языки программирования C и Assembler, с помощью которых очень удобно спроектировать исполнение игры Змейка. В качестве среды разработки была использована программа Microsoft Visual Studio 2019.

Для начала была разработана заставка игры «Змейка». Здесь нас приветствует сама игра и дается информация о создателе. После были созданы переменные X и Y для последующего отображения количества очков, которые пользователь будет получать в течение самой игры.

После чего в модуле Source.cpp начинается сам процесс игры. После запуска начинается проектирование игрового поля, самой змейки, ее цвет, цвет поля. Мы устанавливаем змейку в середину игрового поля. Далее создаем цикл, который ставит координаты яблока случайным образом, но, чтобы они не совпадали со "змейкой".

Движение змейки основано на перемещении в четыре разные стороны. Считывается клавиша, которая была нажата. Если это клавиша направления, то мы выбираем соответствующее направление по координатам, и начинаем движение со смещением на 1, в зависимости от стороны направления. Координаты змейки и каждого её сегмента тела, постоянно считываются и сравниваются с координатами фрукта и самой змейки.

Если координаты головы совпали с координатой фрукта, то к змейке мы прибавляем 1 сегмент тела, а фрукт генерируем в другом месте.

Если координаты головы совпали с самой змейкой, то в функции, которая отвечает за движение, то игра завершается.

В процессе, когда фрукт будет поглощен змейкой, пользователь зарабатывает 1 очко за каждый фрукт. Информация об этом выводится на экран. Результат записывается в файл, и после будет выводиться в таблице лидеров.

Схема работы программы представлена на рисунке 3.

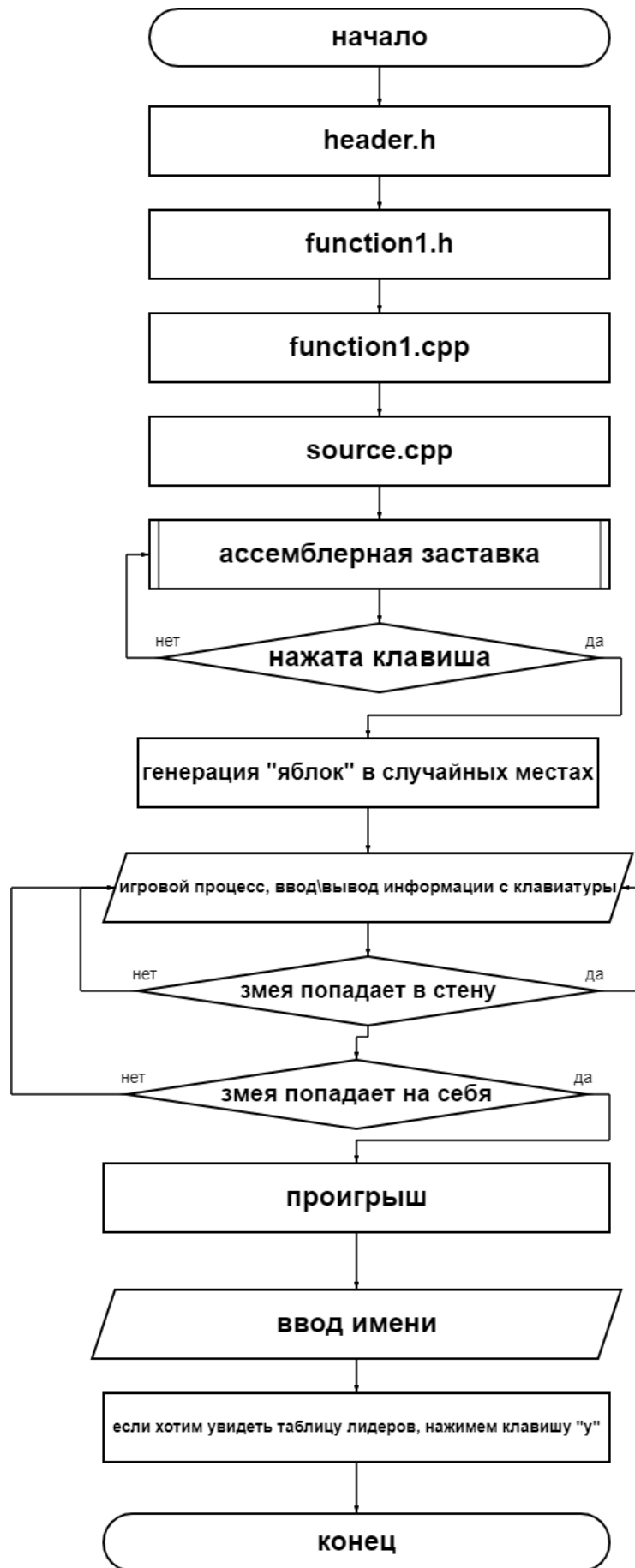


Рисунок 2 – Схема программы

4 Отладка и тестирование

Согласно заданию в качестве среды разработки была выбрана программа Microsoft Visual Studio 2019, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ. Для отладки программы использовались точки останова и пошаговое выполнение кода программы, анализ содержимого глобальных и локальных переменных.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В процессе отладки было выявлено и исправлено множество ошибок, например, некорректное открытие игры при нажатии кнопки «Старт», проблемы связанные с движением змейки (движение не в ту сторону, отказ от движения), проблемы с assembler-ной вставкой, некорректное отображение счета и вывода таблицы лидеров.

5 Описание программы

При запуске программы появляется заставка, в которой игра нас приветствует. Заставка написана на языке Assembler.

Далее нажимаем любую клавишу и происходит переход в саму игру. В ней отрисовывается заранее установленное по размерам игровое поле. После чего сама змейка размещается в середине игрового поля и начинает игру, движением змейки вправо ожидая от пользователя ввода клавиши управления, для изменения направления движения змейки.

В процессе движения считываются координаты змейки и сравниваются положением фрукта и самой змейки. Если они совпадают, то в первом случае змейка увеличивается, а во втором игра завершается. Если змейка врежется в саму себя, то игра завершается.

Помимо этого, осуществляется подсчет очков. За каждое собранное «яблоко» добавляется +1 балл. После чего баллы будут использоваться в таблице лидеров.

При завершении игры выводится надпись «you lose», после чего мы нажимаем любую клавишу и нас просят ввести свое имя. Это имя и ваши очки заносятся в таблицу лидеров. Далее, программа предлагает посмотреть саму таблицу лидеров и тех, кто в ней записан. Сделать это можно при нажатии клавиши «Y», после чего сама таблица будет выведена на экран. Далее нажимаем любую клавишу, и программа завершается.

6 Руководство пользователя

При первом запуске появляется заставка игры. Здесь нас игра приветствует и предлагает нажать любую кнопку для продолжения.

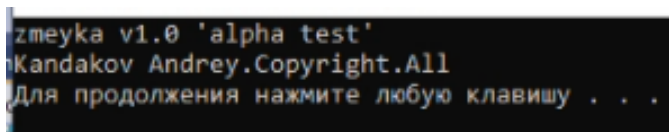


Рисунок 3 – Начало игры

Главное меню появляется после нажатия кнопки, в нем пользователь может начать игру, перейти в другие вкладки, изменить цвет, либо выйти из игры.

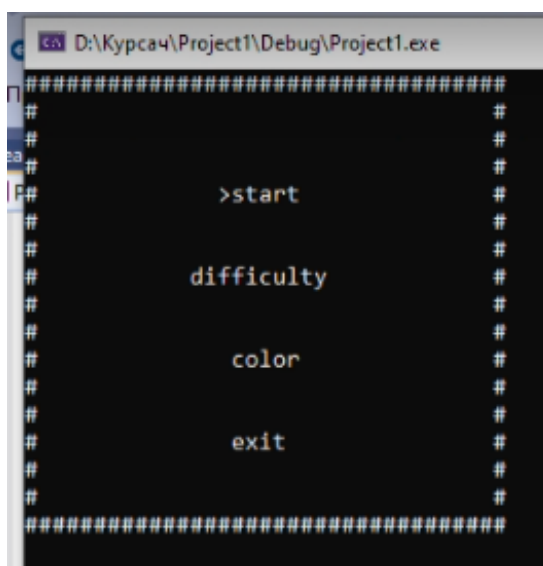


Рисунок 4 – Меню

Пункт выбора сложности на выбор дается 3 варианта.

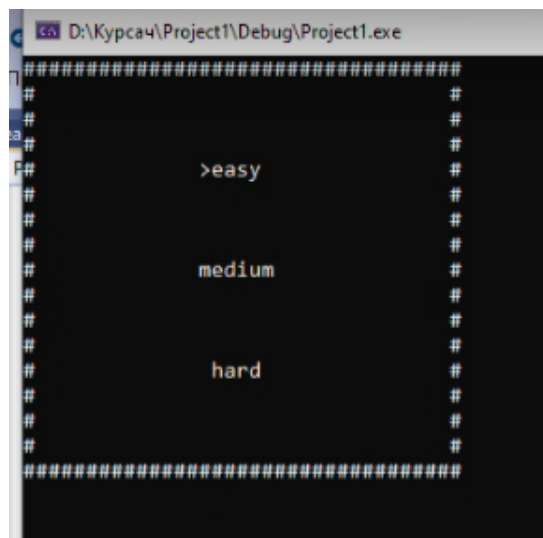


Рисунок 5 – Сложность

Пункт выбора цвета на выбор дается 3 варианта.

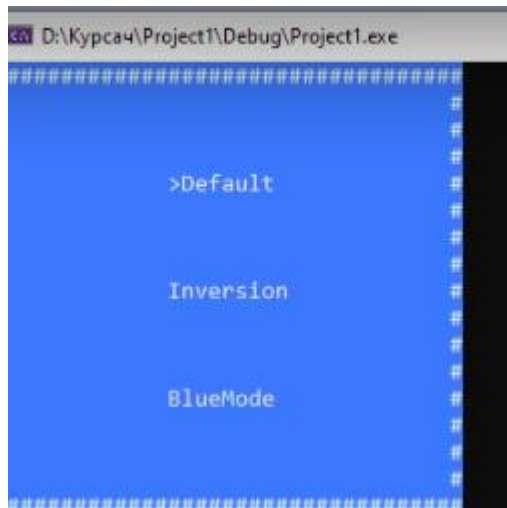


Рисунок 6 – Цвет

После нажатия «Start» у нас запускается сама игра.

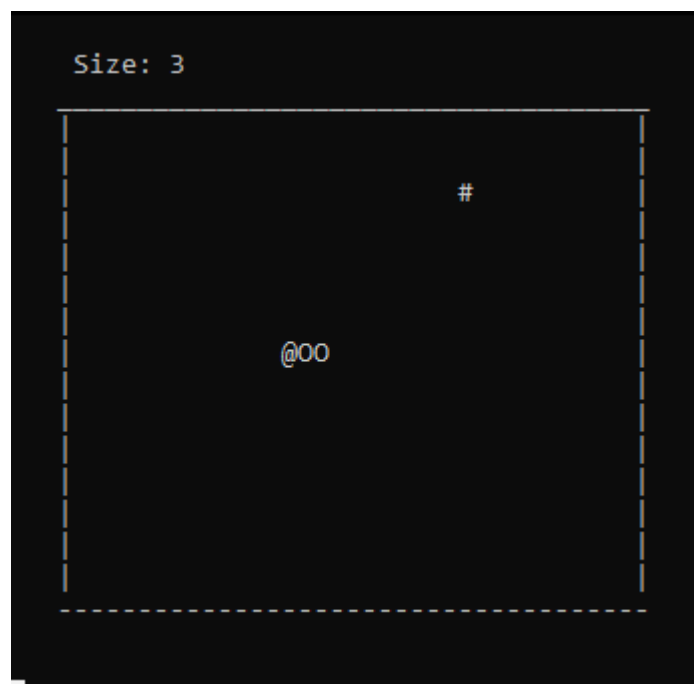


Рисунок 7 – Игровой процесс

С помощью кнопок управления необходимо перемещать змейку так, чтобы она съела появляющийся «фрукт».

A	W	D	S
Движение влево	Движение Вверх	Движение Вправо	Движение Вниз

Рисунок 8 – Управление в игре

После того, как змейка «скушает яблоко», пользователь получает +1 балл, а змейка увеличивается на 1 сегмент.

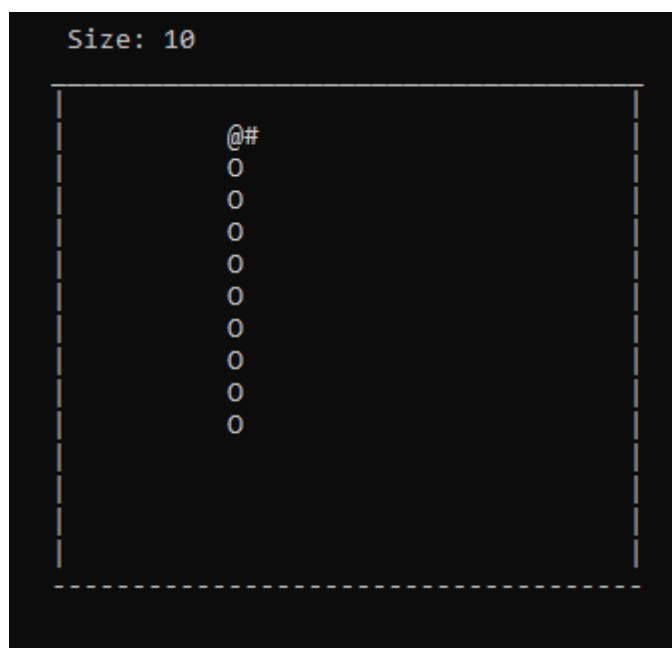


Рисунок 9 – Увеличение змейки и счета

Если змейка врзается в себя, то выводится сообщение о счете и об окончании игры.

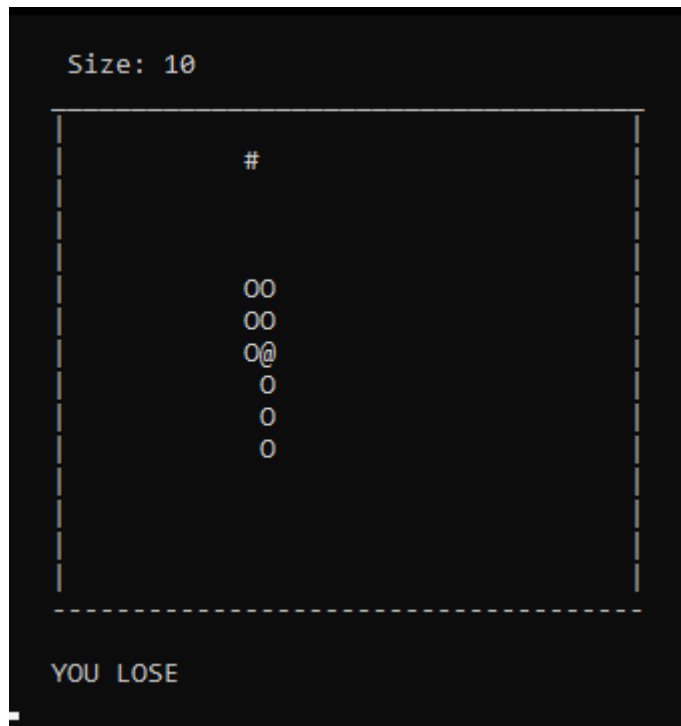


Рисунок 10 – Показ счета и завершение игры

После этого, программа просит ввести свое имя, вводим:

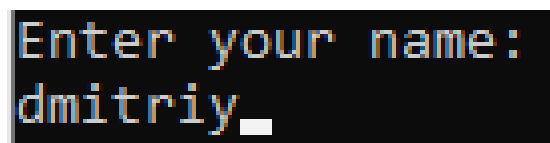


Рисунок 11 – Ввод имени пользователя

Далее, если мы хотим увидеть таблицу лидеров, нажимаем клавишу «Y».

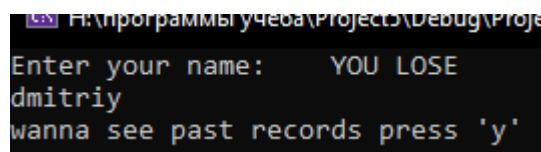


Рисунок 12 – Предложение показа таблицы рекордов

После чего нам показывается таблица лидеров со следующей информацией: имя, количество очков, дата и время, когда игра завершилась.

```
Player Name :LOL
Played Date:Fri May 31 10:28:10 2024
Score:14

Level:10

Player Name :LOL
Played Date:Fri May 31 10:29:18 2024
Score:8

Level:10

Player Name :Lol
Played Date:Fri May 31 10:32:31 2024
Score:20

Level:10

Player Name :Lol
Played Date:Fri May 31 10:33:14 2024
Score:11

Level:10

Для продолжения нажмите любую клавишу . . .
```

Рисунок 13 – Таблица рекордов

Заключение

П В процессе выполнения данной курсовой работы были не только улучшены базовые навыки программирования на языках C и Assembler, но и получен ряд новых знаний и практических умений:

Изучены основные возможности среды программирования Microsoft Visual Studio 2019. Были освоены принципы работы с различными инструментами и функциями этой популярной среды разработки, что открывает возможности для реализации более сложных проектов в будущем. Были освоены способы обработки событий с помощью клавиатуры. Это позволило реализовать интерактивный игровой процесс, в котором игрок может управлять объектами игры с помощью клавиатуры.

Получены навыки отладки и тестирования программ. Были изучены и применены различные методы поиска и устранения ошибок, что является важным навыком для любого программиста.

Были освоены приемы создания и подключения модулей программы. Это позволило разделить код на отдельные модули, что делает программу более организованной и удобной для поддержки и расширения.

Получены навыки работы со сложными типами данных. Были изучены и применены структуры данных, такие как массивы и списки, что позволило эффективно организовать и обрабатывать данные в игре. В результате был разработан рабочий прототип игры «Змейка», обладающей следующими характеристиками: Приятное оформление и простой для использования интерфейс. Игра имеет привлекательный внешний вид и понятный интерфейс, что делает ее удобной и увлекательной для игроков. Функциональный игровой процесс. Игра реализует все основные игровые механизмы классической игры «Змейка», обеспечивая хороший игровой опыт.

Список используемых источников

- 1 Зубков, С. В. Assembler для DOS, Windows и UNIX / Зубков Сергей Владимирович. – 3-е изд., стер. – М. : ДМК Пресс ; СПб. : Питер, 2004. – 608 с. : ил. – (Серия «Для программистов»). – URL: <http://lshoshia.science.tsu.ge/assembler.pdf> (дата обращения: 07.05.2022).
- 2 Рацеев, С. М. Программирование на языке Си : учеб. пособие / С. М. Рацеев. – Ульяновск: УлГУ, 2015. – 319 с. – URL: <https://www.ulsu.ru/media/uploads/ratseevsm%40mail.ru/2019/05/05/Рацеев%20С.М.%20Программирование%20на%20языке%20Си.pdf> (дата обращения: 27.04.2022).
- 3 Солдатенко, И.С. Основы программирования на языке Си: учеб. пособие / И.С. Солдатенко. – Тверь: Твер. гос. ун-т. – 2017. – 159 с. – URL: <http://prog.tversu.ru/library/C.pdf> (дата обращения: 18.04.2022).
- 4 CyberForum.ru. – URL: <https://www.cyberforum.ru/> (дата обращения: 02.05.2022).
- 5 Itrobo.ru. Образовательный портал по программированию и робототехнике. – URL: <http://itrobo.ru/programmirovanie/praktikum/upravlenie-dvizheniem-obekta-na-si.html> (дата обращения: 24.05.2022).
- 6 Stack Overflow. – URL: <https://stackoverflow.com/> (дата обращения: 04.04.2022).
- 7 Stack Overflow на русском. – URL: <https://ru.stackoverflow.com/> (дата обращения: 04.04.2022).

Приложение А

Листинги программы

Приложение А.1

Файл «Header.h»

```
#pragma once
#include <iostream>
#include <thread>
#include <string>
#include <conio.h>
#include <windows.h>
#include <ctime>

using namespace std;

// Выбор пользователя
#define UC_START 0
#define UC_DIFFICULTY 1
#define UC_COLOR 2
#define UC_EXIT 3

// Спящий режим
#define ST_EASY 140
#define ST_MEDIUM 70
#define ST_HARD 14

// Варианты цвета
#define CS_DEFAULT 0x000F

#define CS_FIELD_PACK1 0x000F
#define CS_SNAKE_PACK1 0x000A
#define CS_APPLE_PACK1 0x000C

#define CS_FIELD_PACK2 0x00F0
#define CS_SNAKE_PACK2 0x00F2
#define CS_APPLE_PACK2 0x00FC

#define CS_FIELD_PACK3 0x009F
#define CS_SNAKE_PACK3 0x009F
#define CS_APPLE_PACK3 0x009C
```

Приложение А.2

Файл «Function1.h»

```
#pragma once  
void setcur(int x, int y);
```

Приложение А.3

Файл «Function1.cpp»

```
#include "Header.h"
void setcur(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),
        coord);
}}
```

Приложение А.4

Файл «Source.cpp»

```
#pragma warning(disable : 4996)
#include "Header.h"
#include "function1.h"

// Глобальные переменные
int sleepTime = ST_EASY;
CRITICAL_SECTION cs;

WORD fieldAttr = CS_FIELD_PACK1;
WORD snakeAttr = CS_SNAKE_PACK1;
WORD appleAttr = CS_APPLE_PACK1;

void record(int* score) {
    CONSOLE_CURSOR_INFO cci;
    GetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cci);
    cci.bVisible = true;
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cci);
    char pname[100], nplname[100], cha, c;
    int i, j;
    FILE* info;
    info = fopen("record.txt", "a+");
    //_getch();
    system("cls");
    printf("Enter your name: ");
    scanf("%s", pname);
    //*****
    for (j = 0; pname[j] != '\0'; j++) { //чтобы преобразовать первую букву
        после пробела в заглавную
        nplname[0] = toupper(pname[0]);
        if (pname[j - 1] == ' ') {
            nplname[j] = toupper(pname[j]);
            nplname[j - 1] = pname[j - 1];
        }
        else nplname[j] = pname[j];
    }
    nplname[j] = '\0';
    //*****
    fprintf(info, "Player Name :%s\n", nplname);
    //для даты и времени
    time_t mytime;
    mytime = time(NULL);
    fprintf(info, "Played Date:%s", ctime(&mytime));
    //*****
    fprintf(info, "Score:%d\n", *score); //вызова для отображения оценки
    fprintf(info, "\nLevel:%d\n", 10); //вызова для отображения уровня
    for (i = 0; i <= 50; i++)
        fprintf(info, "%c", '_');
    fprintf(info, "\n");
    fclose(info);
}
```

```

printf("wanna see past records press 'y'\n");
cha = _getch();
system("cls");
if (cha == 'y') {
    info = fopen("record.txt", "r");
    do {
        putchar(c = getc(info));
    } while (c != EOF);
}
fclose(info);
}

int zmeia(int width, int heigth, int* bodyCoors[2], char** charField,
string kbInput[1], int snakeLen, int* score)
{
    int m, n;
    char cEmpty = ' ';
    bool lose = false, Win = false;
    int savedLen = snakeLen;

    while ((lose == 0) && (Win == 0))
    {
        EnterCriticalSection(&cs); // Вход в критическу секцию

        // Сдвиг координат тела змейки
        for (int i = snakeLen - 1; i > 0; i--)
        {
            bodyCoors[0][i] = bodyCoors[0][i - 1];
            bodyCoors[1][i] = bodyCoors[1][i - 1];
        }

        if (((kbInput[0] == "a") || (kbInput[0] == "A"))) // Управление с
клавиатуры
        {
            if (bodyCoors[0][0] == 0)
            {
                bodyCoors[0][0] = width;
            }
            bodyCoors[0][0] = bodyCoors[0][0] - 1;

            if (charField[bodyCoors[1][0]][bodyCoors[0][0]] == '#')
            {
                int appleY, appleX;
                snakeLen++;

                while (true)
                {
                    appleY = rand() % heigth;
                    appleX = rand() % width;
                    if ((charField[appleY][appleX] == cEmpty) ||
(snakeLen == width * heigth))
                    {

```

```

        if (snakeLen != width * height)
charField[appleY][appleX] = '#';
        break;
    }
}
bodyCoors[0][savedLen] = 1000;
bodyCoors[1][savedLen] = 1000;
}
else
if (((kbInput[0] == "d") || (kbInput[0] == "D")))
{
    if (bodyCoors[0][0] == width - 1)
    {
        bodyCoors[0][0] = -1;
    }
    bodyCoors[0][0] = bodyCoors[0][0] + 1;

    if (charField[bodyCoors[1][0]][bodyCoors[0][0]] == '#')
    {
        snakeLen++;
        int appleY, appleX;
        while (true)
        {
            appleY = rand() % height;
            appleX = rand() % width;
            if ((charField[appleY][appleX] == cEmpty) ||
(snakeLen == width * height))
            {
                if (snakeLen != width * height)
charField[appleY][appleX] = '#';
                break;
            }
        }
    }
    bodyCoors[0][savedLen] = 1000;
    bodyCoors[1][savedLen] = 1000;
}
else
if (((kbInput[0] == "w") || (kbInput[0] == "W")))
{
    // Проверка координат на верхней границе поля
    if (bodyCoors[1][0] == 0)
    {
        bodyCoors[1][0] = height;
    }
    bodyCoors[1][0]--;

    if (charField[bodyCoors[1][0]][bodyCoors[0][0]] == '#')
    {
        snakeLen++;
        int appleY, appleX;
        while (true)

```

```

        {
            appleY = rand() % height;
            appleX = rand() % width;
            if ((charField[appleY][appleX] == cEmpty) ||
(snakeLen == width * height))
            {
                if (snakeLen != width * height)
charField[appleY][appleX] = '#';
                break;
            }
        }
        bodyCoors[0][savedLen] = 1000;
        bodyCoors[1][savedLen] = 1000;
    }
    else if (((kbInput[0] == "s") || (kbInput[0] == "S")))
    {
        if (bodyCoors[1][0] == height - 1)
        {
            bodyCoors[1][0] = -1;
        }
        bodyCoors[1][0] = bodyCoors[1][0] + 1;

        if (charField[bodyCoors[1][0]][bodyCoors[0][0]] == '#')
        {
            snakeLen++;
            int appleY, appleX;
            while (true)
            {
                appleY = rand() % height;
                appleX = rand() % width;
                if ((charField[appleY][appleX] == cEmpty) ||
(snakeLen == width * height))
                {
                    if (snakeLen != width * height)
charField[appleY][appleX] = '#';
                    break;
                }
            }
            bodyCoors[0][savedLen] = 1000;
            bodyCoors[1][savedLen] = 1000;
        }

        // Проверка координат головы змейки на идентичность с координатами ячеек тела
        for (int i = 1; i < snakeLen; i++)
        {
            if ((bodyCoors[0][0] == bodyCoors[0][i]) && (bodyCoors[1][0]
== bodyCoors[1][i]))
            {
                lose++;
                kbInput[0] = "vse";
            }
        }
    }
}

```

```

        break;
    }
}

if (snakeLen == width * height)
{
    kbInput[0] = "vse";
    LeaveCriticalSection(&cs);

    break;
}

savedLen = snakeLen;

for (int i = 0; i < height; i++)
{
    for (int j = 0; j < width; j++)
    {
        if (charField[i][j] != '#') charField[i][j] = cEmpty;
    }
}

for (int i = 0; i < snakeLen; i++)
{
    if ((bodyCoors[0][i] != 1000) && (bodyCoors[1][i] != 1000))
    {
        m = bodyCoors[0][i];
        n = bodyCoors[1][i];
        charField[n][m] = 'O';
    }
}

m = bodyCoors[0][0];
n = bodyCoors[1][0];

charField[n][m] = '@';

// Рисование поля с объектами
SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
fieldAttr);

setcur(0, 1);
for (int i = 0; i < width + 3; i++) cout << " ";

setcur(0, 1);
cout << " Size: " << snakeLen;
cout << "\n ";
for (int i = 0; i < width + 2; i++) cout << "_";
cout << "\n";

for (int i = 0; i < height; i++)
{
    cout << " |";

```

```

        for (int j = 0; j < width; j++)
        {
            if (charField[i][j] == '@' || charField[i][j] == 'O')
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), snakeAttr);
            else if (charField[i][j] == '#')
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), appleAttr);
            else
                SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), fieldAttr);

            cout << charField[i][j];
        }
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
fieldAttr);
        cout << "|" << endl;
    }
    cout << " ";
    for (int i = 0; i < width + 2; i++) cout << "-";
    cout << "\n";

    // Задержка. Влияет на скорость игры
    Sleep(sleepTime);

    if (lose) { LeaveCriticalSection(&cs); break; }

    LeaveCriticalSection(&cs);
    Sleep(5);
}

cout << " YOU " << (!lose ? "WIN\n" : "LOSE\n");

SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), CS_DEFAULT);

kbInput[0] = "vse";
*score = snakeLen;

return 0;
}

```

```

int vvod(string c[1])
{
    char C = '\0';
    while (c[0] != "vse") {
        C = _getch();

        if (C != 'w' && C != 'W' &&
            C != 'a' && C != 'A' &&
            C != 's' && C != 'S' &&
            C != 'd' && C != 'D')
            continue;
    }
}

```

```

        EnterCriticalSection(&cs);

        if (((c[0] == "d") || (c[0] == "D")) && !((C == 'a') || (C ==
'A'))))
        {
            c[0] = C;
        }
        else
        if (((c[0] == "a") || (c[0] == "A")) && !((C == 'd') || (C ==
'D'))))
        {
            c[0] = C;
        }
        else
        if (((c[0] == "w") || (c[0] == "W")) && !((C == 's') || (C ==
'S'))))
        {
            c[0] = C;
        }
        else
        if (((c[0] == "s") || (c[0] == "S")) && !((C == 'w') || (C ==
'W'))))
        {
            c[0] = C;
        }

        LeaveCriticalSection(&cs);
        Sleep(10);
    }
    return 0;
}

```

```

void casino() {
    char word_1[] = "zmeyka v1.0 'alpha test'\n" "Kandakov
Andrey.Copyright.All\n";
    __asm {
        lea eax, word_1;
        push eax;
        call printf;
        add esp, 4;
    }
    system("pause");
    system("cls");
}

```

```

void printFrame(int width, int heigth) {
    setcur(0, 0);

    for (int i = 0; i < width; i++) {
        printf("#");
    }
}

```

```

    }

    printf("\n");

    for (int i = 1; i < heigth - 1; i++) {
        for (int j = 0; j < width; j++) {
            if (j == 0 || j == width - 1) printf("#");
            else printf(" ");
        }
        printf("\n");
    }

    for (int i = 0; i < width; i++) {
        printf("#");
    }
}

int difficultyScreen(int state, int* width, int* heigth) {
    int nUserChoice = state;
    int clientWidth = 35 - 2, clientHeigth = 17 - 2;
    char easy[] = "easy";
    char medium[] = "medium";
    char hard[] = "hard";
    char c = '\0';

    setcur(clientWidth / 2 + 1 - strlen(easy) / 2, clientHeigth / 4 + 1);
    printf("%s", easy);

    setcur(clientWidth / 2 + 1 - strlen(medium) / 2, clientHeigth / 2 +
1);
    printf("%s", medium);

    setcur(clientWidth / 2 + 1 - strlen(hard) / 2, clientHeigth * 3 / 4 +
1);
    printf("%s", hard);

    while (true) {
        setcur(clientWidth / 2 - strlen(easy) / 2, clientHeigth / 4 + 1);
        printf(" ");
        setcur(clientWidth / 2 - strlen(medium) / 2, clientHeigth / 2 +
1);
        printf(" ");
        setcur(clientWidth / 2 - strlen(hard) / 2, clientHeigth * 3 / 4 +
1);
        printf(" ");

        switch (nUserChoice)
        {
            case 0:
                setcur(clientWidth / 2 - strlen(easy) / 2, clientHeigth / 4 +
1);
                printf(">");

```

```

        break;
    case 1:
        setcur(clientWidth / 2 - strlen(medium) / 2, clientHeigth / 2
+ 1);
        printf(">");
        break;
    case 2:
        setcur(clientWidth / 2 - strlen(hard) / 2, clientHeigth * 3 /
4 + 1);
        printf(">");
        break;
    default:
        break;
}

c = _getch();

if (c == -32) {
    c = _getch();
    if (c == 0x48) {
        nUserChoice = nUserChoice == 0 ? 2 : nUserChoice - 1;
    }
    else if (c == 0x50) {
        nUserChoice = nUserChoice == 2 ? 0 : nUserChoice + 1;
    }
}
else if (c == 13) {
    setcur(clientWidth / 2 - strlen(easy) / 2, clientHeigth / 4 +
1);

    for (int i = 0; i < strlen(easy) + 1; i++) printf(" ");
    setcur(clientWidth / 2 - strlen(medium) / 2, clientHeigth / 2
+ 1);

    for (int i = 0; i < strlen(medium) + 1; i++) printf(" ");
    setcur(clientWidth / 2 - strlen(hard) / 2, clientHeigth * 3 /
4 + 1);

    for (int i = 0; i < strlen(hard) + 1; i++) printf(" ");

    switch (nUserChoice)
    {
    case 0:
        *width = 10;
        *height = 5;
        return ST_EASY;
        break;
    case 1:
        *width = 20;
        *height = 10;
        return ST_MEDIUM;
        break;
    case 2:
        *width = 35;
        *height = 17;
        return ST_HARD;
    }
}

```

```

        break;
    default:
        break;
    }
}
}

int ColorScreen(int state) {
    int nUserChoice = state;
    int clientWidth = 35 - 2, clientHeight = 17 - 2;
    char cDefault[] = "Default";
    char cInversion[] = "Inversion";
    char cBlue[] = "BlueMode";
    char c = '\\0';

    setcur(clientWidth / 2 + 1 - strlen(cDefault) / 2, clientHeight / 4 +
1);
    printf("%s", cDefault);

    setcur(clientWidth / 2 + 1 - strlen(cInversion) / 2, clientHeight / 2
+ 1);
    printf("%s", cInversion);

    setcur(clientWidth / 2 + 1 - strlen(cBlue) / 2, clientHeight * 3 / 4
+ 1);
    printf("%s", cBlue);

    while (true) {
        setcur(clientWidth / 2 - strlen(cDefault) / 2, clientHeight / 4 +
1);
        printf(" ");
        setcur(clientWidth / 2 - strlen(cInversion) / 2, clientHeight / 2
+ 1);
        printf(" ");
        setcur(clientWidth / 2 - strlen(cBlue) / 2, clientHeight * 3 / 4
+ 1);
        printf(" ");

        switch (nUserChoice)
        {
            case 0:
                setcur(clientWidth / 2 - strlen(cDefault) / 2, clientHeight /
4 + 1);
                printf(">");
                break;
            case 1:
                setcur(clientWidth / 2 - strlen(cInversion) / 2, clientHeight
/ 2 + 1);
                printf(">");
                break;
            case 2:

```

```

        setcur(clientWidth / 2 - strlen(cBlue) / 2, clientHeigth * 3
/ 4 + 1);
        printf(">");
        break;
default:
        break;
}

c = _getch();

if (c == -32) {
    c = _getch();
    if (c == 0x48) {
        nUserChoice = nUserChoice == 0 ? 2 : nUserChoice - 1;
    }
    else if (c == 0x50) {
        nUserChoice = nUserChoice == 2 ? 0 : nUserChoice + 1;
    }
}
else if (c == 13) {
    setcur(clientWidth / 2 - strlen(cDefault) / 2, clientHeigth /
4 + 1);

    for (int i = 0; i < strlen(cDefault) + 1; i++) printf(" ");
    setcur(clientWidth / 2 - strlen(cInversion) / 2, clientHeigth
/ 2 + 1);

    for (int i = 0; i < strlen(cInversion) + 1; i++) printf(" ");
    setcur(clientWidth / 2 - strlen(cBlue) / 2, clientHeigth * 3
/ 4 + 1);

    for (int i = 0; i < strlen(cBlue) + 1; i++) printf(" ");

    switch (nUserChoice)
    {
    case 0:
        fieldAttr = CS_FIELD_PACK1;
        snakeAttr = CS_SNAKE_PACK1;
        appleAttr = CS_APPLE_PACK1;
        return 0;
        break;
    case 1:
        fieldAttr = CS_FIELD_PACK2;
        snakeAttr = CS_SNAKE_PACK2;
        appleAttr = CS_APPLE_PACK2;
        return 1;
        break;
    case 2:
        fieldAttr = CS_FIELD_PACK3;
        snakeAttr = CS_SNAKE_PACK3;
        appleAttr = CS_APPLE_PACK3;
        return 2;
        break;
    default:
        break;
    }
}

```

```

    }
}

int menuScreen(int* width, int* height) {
    int nUserChoice = 0;
    int colorState = 0, difficultyState = 0;
    int clientWidth = 35 - 2, clientHeight = 17 - 2;
    char start[] = "start";
    char difficulty[] = "difficulty";
    char color[] = "color";
    char exit[] = "exit";
    char c = '\0';

    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), fieldAttr);

    printFrame(35, 17);

    setcur(clientWidth / 2 + 1 - strlen(start) / 2, clientHeight / 5 +
1);
    printf("%s", start);

    setcur(clientWidth / 2 + 1 - strlen(difficulty) / 2, clientHeight * 2
/ 5 + 1);
    printf("%s", difficulty);

    setcur(clientWidth / 2 + 1 - strlen(color) / 2, clientHeight * 3 / 5
+ 1);
    printf("%s", color);

    setcur(clientWidth / 2 + 1 - strlen(exit) / 2, clientHeight * 4 / 5 +
1);
    printf("%s", exit);

    while (true) {
        setcur(clientWidth / 2 - strlen(start) / 2, clientHeight / 5 +
1);
        printf(" ");
        setcur(clientWidth / 2 - strlen(difficulty) / 2, clientHeight * 2
/ 5 + 1);
        printf(" ");
        setcur(clientWidth / 2 - strlen(exit) / 2, clientHeight * 3 / 5 +
1);
        printf(" ");
        setcur(clientWidth / 2 - strlen(exit) / 2, clientHeight * 4 / 5 +
1);
        printf(" ");

        switch (nUserChoice)
        {
            case UC_START:
                setcur(clientWidth / 2 - strlen(start) / 2, clientHeight / 5

```

```

+ 1);
        printf(">");
        break;
    case UC_DIFICULTY:
        setcur(clientWidth / 2 - strlen(difficulty) / 2, clientHeigh
* 2 / 5 + 1);
        printf(">");
        break;
    case UC_COLOR:
        setcur(clientWidth / 2 - strlen(color) / 2, clientHeigh * 3
/ 5 + 1);
        printf(">");
        break;
    case UC_EXIT:
        setcur(clientWidth / 2 - strlen(exit) / 2, clientHeigh * 4 /
5 + 1);
        printf(">");
        break;
    default:
        break;
}

c = _getch();

if (c == -32) {
    c = _getch();
    if (c == 0x48) {
        nUserChoice = nUserChoice == UC_START ? UC_EXIT :
nUserChoice - 1;
    }
    else if (c == 0x50) {
        nUserChoice = nUserChoice == UC_EXIT ? UC_START :
nUserChoice + 1;
    }
}
else if (c == 13) {
    switch (nUserChoice)
    {
    case UC_START:
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
CS_DEFAULT);
        system("cls");
        return nUserChoice;
        break;
    case UC_DIFICULTY:
        setcur(clientWidth / 2 - strlen(start) / 2, clientHeigh
/ 5 + 1);
        for (int i = 0; i < strlen(start) + 1; i++) printf(" ");
        setcur(clientWidth / 2 - strlen(difficulty) / 2,
clientHeigh * 2 / 5 + 1);
        for (int i = 0; i < strlen(difficulty) + 1; i++) printf("
");
        setcur(clientWidth / 2 - strlen(color) / 2, clientHeigh

```

```

* 3 / 5 + 1);
    for (int i = 0; i < strlen(color) + 1; i++) printf(" ");
    setcur(clientWidth / 2 - strlen(exit) / 2, clientHeigh *
4 / 5 + 1);
    for (int i = 0; i < strlen(exit) + 1; i++) printf(" ");

    sleepTime = difficultyScreen(difficultyState, width,
height);

    switch (sleepTime)
    {
    case ST_EASY:
        difficultyState = 0;
        break;
    case ST_MEDIUM:
        difficultyState = 1;
        break;
    case ST_HARD:
        difficultyState = 2;
        break;
    default:
        break;
    }

    setcur(clientWidth / 2 + 1 - strlen(start) / 2,
clientHeigh / 5 + 1);
    printf("%s", start);

    setcur(clientWidth / 2 + 1 - strlen(difficulty) / 2,
clientHeigh * 2 / 5 + 1);
    printf("%s", difficulty);

    setcur(clientWidth / 2 + 1 - strlen(color) / 2,
clientHeigh * 3 / 5 + 1);
    printf("%s", color);

    setcur(clientWidth / 2 + 1 - strlen(exit) / 2,
clientHeigh * 4 / 5 + 1);
    printf("%s", exit);
    break;
case UC_COLOR:
    setcur(clientWidth / 2 - strlen(start) / 2, clientHeigh
/ 5 + 1);
    for (int i = 0; i < strlen(start) + 1; i++) printf(" ");
    setcur(clientWidth / 2 - strlen(difficulty) / 2,
clientHeigh * 2 / 5 + 1);
    for (int i = 0; i < strlen(difficulty) + 1; i++) printf("
");
    setcur(clientWidth / 2 - strlen(color) / 2, clientHeigh
* 3 / 5 + 1);
    for (int i = 0; i < strlen(color) + 1; i++) printf(" ");
    setcur(clientWidth / 2 - strlen(exit) / 2, clientHeigh *
4 / 5 + 1);

```

```

        for (int i = 0; i < strlen(exit) + 1; i++) printf(" ");

        colorState = ColorScreen(colorState);

        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
fieldAttr);

        printFrame(35, 17);

        setcur(clientWidth / 2 + 1 - strlen(start) / 2,
clientHeigth / 5 + 1);
        printf("%s", start);

        setcur(clientWidth / 2 + 1 - strlen(difficulty) / 2,
clientHeigth * 2 / 5 + 1);
        printf("%s", difficulty);

        setcur(clientWidth / 2 + 1 - strlen(color) / 2,
clientHeigth * 3 / 5 + 1);
        printf("%s", color);

        setcur(clientWidth / 2 + 1 - strlen(exit) / 2,
clientHeigth * 4 / 5 + 1);
        printf("%s", exit);
        break;
    case UC_EXIT:
        return nUserChoice;
        break;
    default:
        break;
    }
}
}
}

```

```

int main()
{
    CONSOLE_CURSOR_INFO cci;

    int width = 10, height = 5;
    int choice = 0;
    char** charField = NULL;
    char cEmpty = ' ';
    int score;
    int* bodyCoors[2];
    int snakeLen = 3;
    int y, x;
    string kbInput[1] = { "a" };

    GetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cci);
    cci.bVisible = false;
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cci);

```

```

InitializeCriticalSection(&cs);

srand(time(0));

casino();
choice = menuScreen(&width, &height);

if (choice == UC_EXIT) {
    return 0;
}

charField = (char**)malloc(sizeof(char*) * height);
for (int i = 0; i < height; i++) charField[i] =
(char*)malloc(sizeof(char) * width);

for (int i = 0; i < height; i++)
{
    for (int j = 0; j < width; j++)
    {
        charField[i][j] = cEmpty;
    }
}

bodyCoors[0] = (int*)malloc(sizeof(int) * height * width);
bodyCoors[1] = (int*)malloc(sizeof(int) * height * width);

bodyCoors[0][0] = width / 2 - 1; bodyCoors[1][0] = height / 2;
bodyCoors[0][1] = width / 2; bodyCoors[1][1] = height / 2;
bodyCoors[0][2] = width / 2 + 1; bodyCoors[1][2] = height / 2;

x = bodyCoors[0][0]; y = bodyCoors[1][0];
charField[y][x] = 'O';
x = bodyCoors[0][1]; y = bodyCoors[1][1];
charField[y][x] = 'O';
x = bodyCoors[0][2]; y = bodyCoors[1][2];
charField[y][x] = 'O';

while (true)
{
    int appleY, appleX;

    appleY = rand() % height;
    appleX = rand() % width;

    if (charField[appleY][appleX] == cEmpty)
    {
        charField[appleY][appleX] = '#';
        break;
    }
}

std::thread thr2(zmeia, width, height, bodyCoors, charField, kbInput,

```

```
snakeLen, &score);  
    thr2.detach();  
    vvod(kbInput);  
  
    DeleteCriticalSection(&cs);  
  
    free(bodyCoors[0]);  
    free(bodyCoors[1]);  
  
    for (int i = 0; i < heigth; i++) free(charField[i]);  
    free(charField);  
  
    record(&score);  
  
    system("pause");  
    return 0;  
}
```

Приложение В

Результаты работы программы

Приложение В.1

Скриншоты работы программы

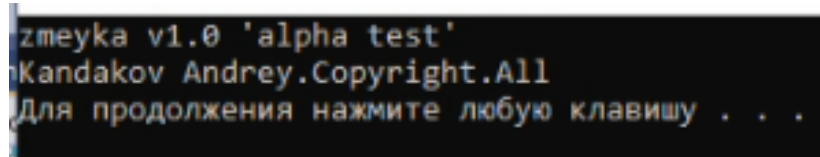


Рисунок 14 – Заставка

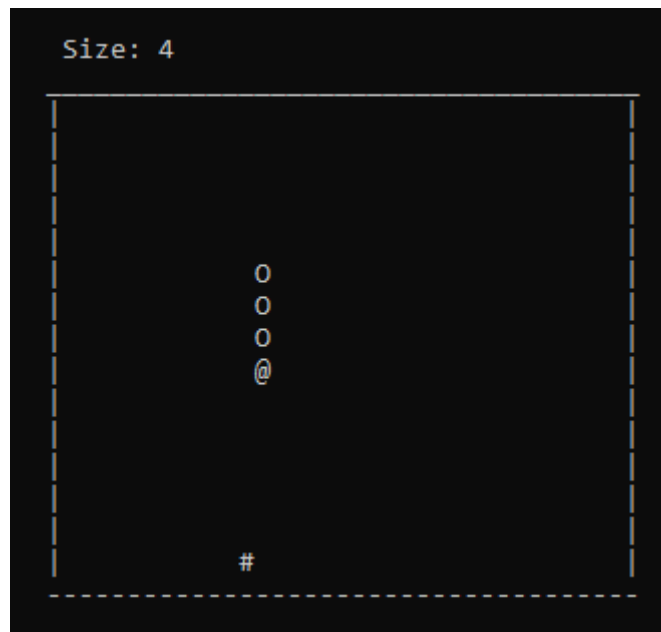


Рисунок 15 – Игровой процесс

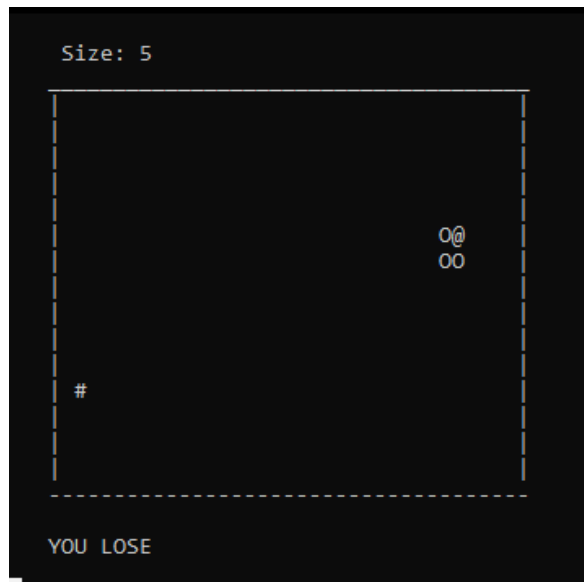


Рисунок 16 – Конец игры

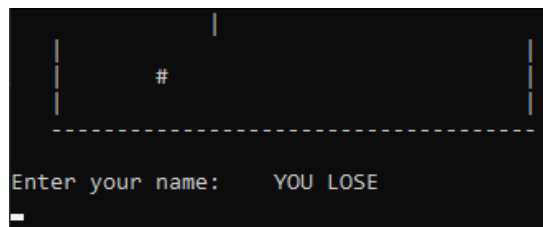


Рисунок 17 – Ввод имени

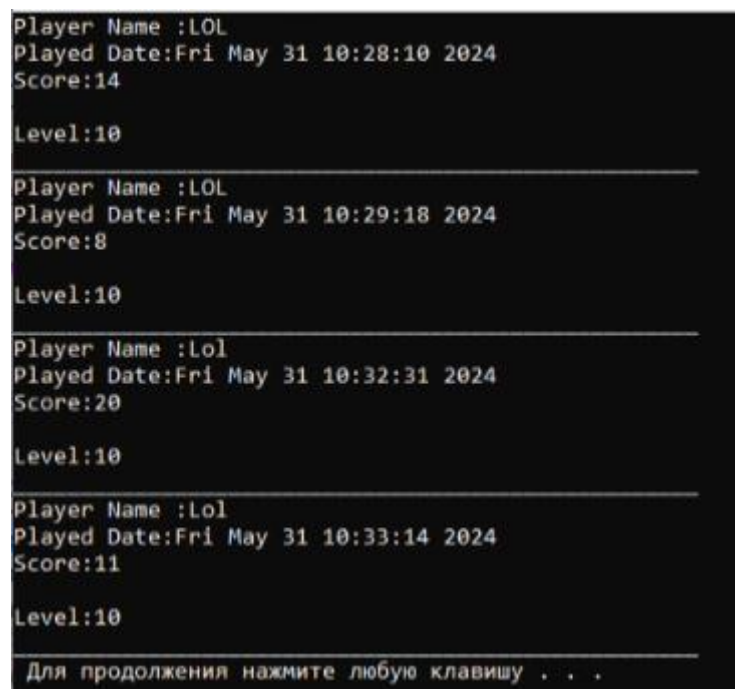


Рисунок 18 – Вывод таблицы лидеров

Приложение В.2

Схемы работы программы

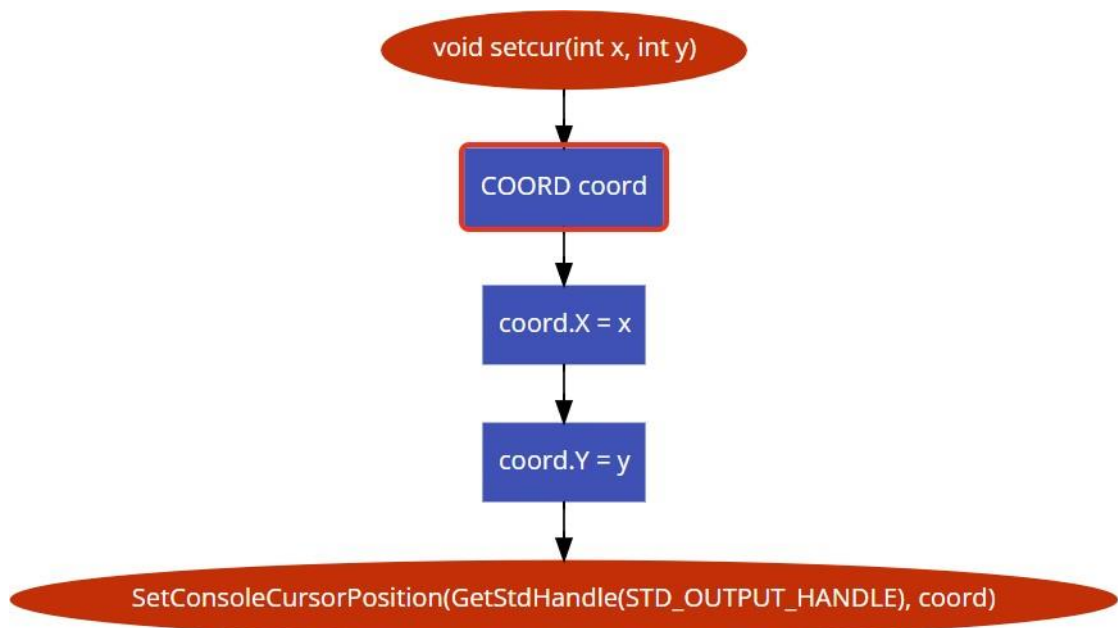


Рисунок 19 – схема абсциссы X и абсциссы Y для вывода данных

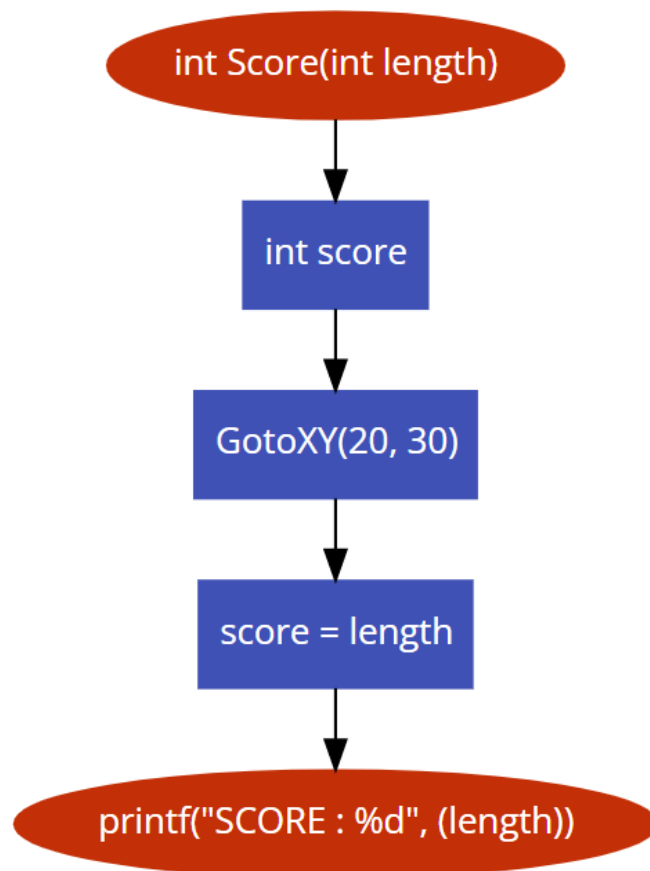


Рисунок 20– схема вывода счета

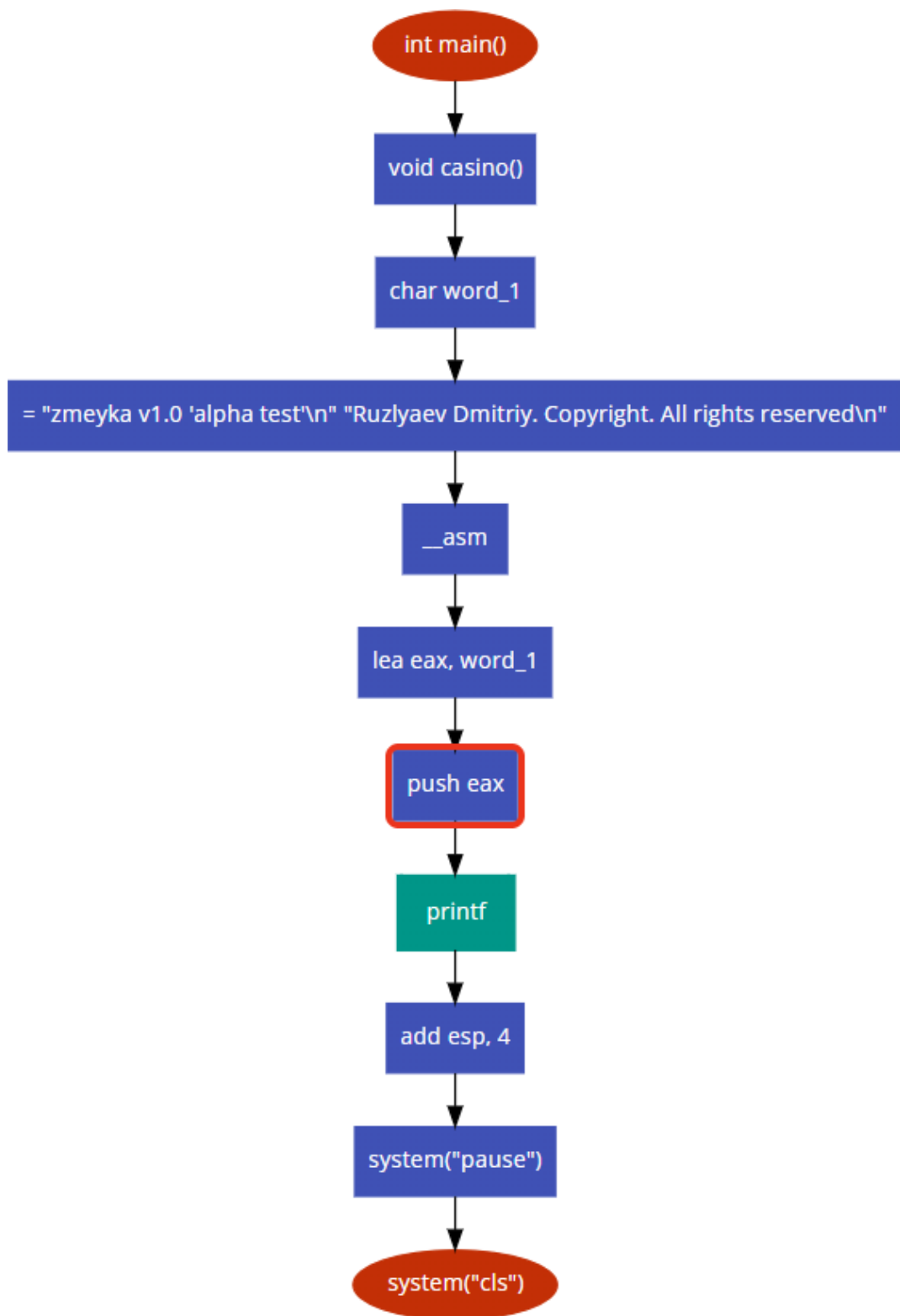


Рисунок 21 – схема вставки на Assembler