

Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему «Игра Змейка»

Выполнил:

студент группы 21BV3

Столяров А.А.

Принял:

Патунин Д.В.

09.06.2022 г.

 Д.В.П.

Пенза, 2022 г.

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 Стоярову Артему Алексеевичу Группа 21ВВЗ

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

Исходные данные (технические требования) на проектирование

- Игра – приложение "Змейка". Функции программы:*
- Управление движением змейки осуществляется с помощью клавиатуры
 - Система набора очков
 - Должен быть предусмотрен рост и уменьшение размеров змейки
 - Добавить таблицу лидеров на 10 мест по очкам.

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" сентября 2022

Дата защиты проекта "05" 06. 2022

Руководитель Дмитрий Васильевич Петушин.

Задание получил "25" сентября 2022 г.

Студент Столяров Артем Алексеевич

Содержание

Введение.....	3
1 Постановка задачи.....	4
2 Выбор решения.....	9
2.1 Определение необходимых модулей программы.....	9
3 Описание разработки программы.....	11
4 Отладка и тестирование.....	13
5 Описание программы.....	14
6 Руководство пользователя.....	16
Список используемых источников.....	21
Приложение А	22
Листинги программы	22
Приложение Б	34

Введение

С развитием технологий компьютеры все больше и больше вливаются в повседневную жизнь человека. Персональные компьютеры – это уже не просто машины для математических вычислений, их общедоступность значительно расширила область применения. И, конечно же, здесь не обошлось без игр.

Индустрия игр быстро набирала обороты, стремительно развивалась, завоевывая все большую и большую популярность у подрастающего поколения. И если первые игры отличались простотой дизайна и логики, то современная компьютерная игра – очень сложная программа, дело рук большого коллектива разработчиков. Так или иначе, компьютерные игры занимают не последнее место в жизни многих современных детей.

Целью этой курсовой работы является создание игры Змейка, одной из самых первых разработанных игры в истории игровой индустрии, а также получение практических навыков написания и разработки приложений с использованием программной среды разработки Microsoft Visual Studio 2022.

1 Постановка задачи

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

- 1 «Змейка» при приближении к границе игрового поля, будет врезаться в него.
- 2 Основной целью игры является увеличении змейки до, как можно большего размера, путем сбора фруктов.
- 3 За каждый собранный фрукт пользователю будут начисляться очки.
- 4 Сам фрукт должен генерироваться в случайном месте.
- 5 Змейка не может проходить через саму себя, поэтому при столкновении со своим туловищем будет наступать конец игры.
- 6 Должна быть создана таблица лидеров.
- 7 Очки должны выводиться сбоку рядом с самой игрой.

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

Необходимо реализовать минималистичный и удобный интерфейс, что бы у пользователя не возникало затруднений при использовании программы.

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

В программе также будут использоваться сложные типы данных.

Устройствами 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;
- Funk.h;
- Sourse.cpp;
- Funk.spp.

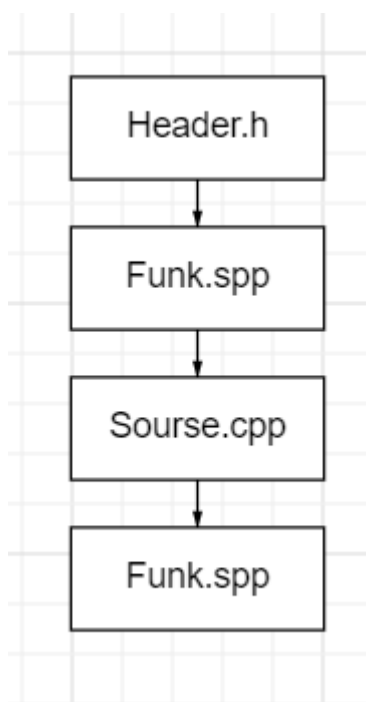


Рисунок 1 – Схема взаимодействия системы

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

Funk.spp – в данном модуле добавляются абсцисса X и абсцисса Y для вывода каких-либо данных в конкретное место в командной строке.

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

Ниже на рисунке 2 представлена схема данных программы.



Рисунок 2 – Схема данных

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 2022.

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

После чего в модуле Source.spp 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 (стрелочки), 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 считываться и сравниваться 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.

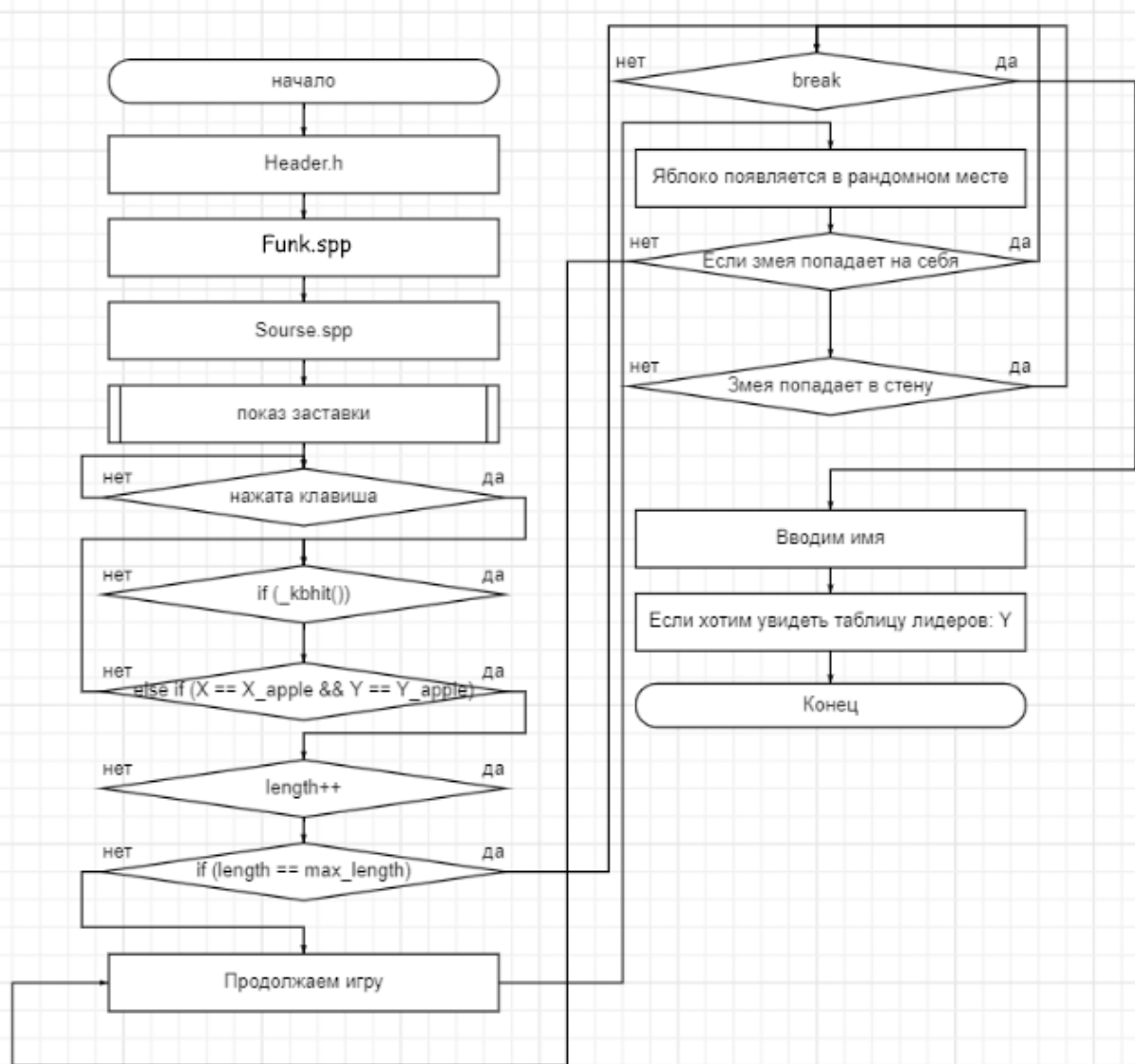


Рисунок 3 – Схема программы

4 Отладка и тестирование

Согласно заданию в качестве среды разработки была выбрана программа Microsoft Visual Studio 2022, 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 каждый собранный фрукт добавляется +1 балл. После чего баллы будут использоваться в таблице лидеров.

При завершении игры выводится надпись «Game over», 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.

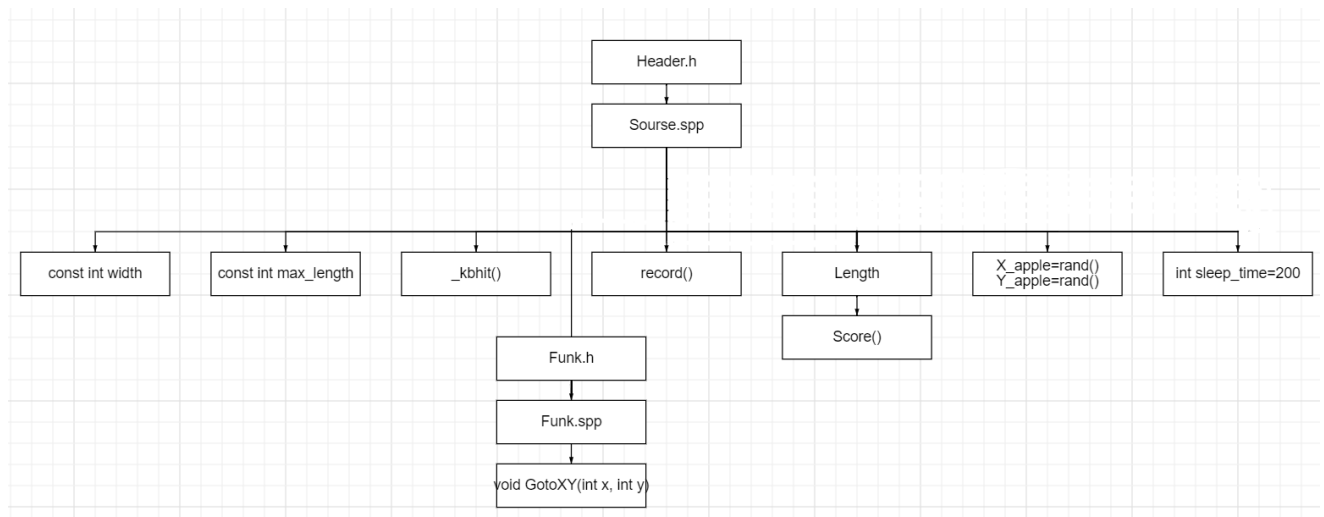


Рисунок 4 – Иерархическая схема программы

6 Руководство пользователя

При первом запуске появляется заставка игры. Здесь нас игра приветствует.

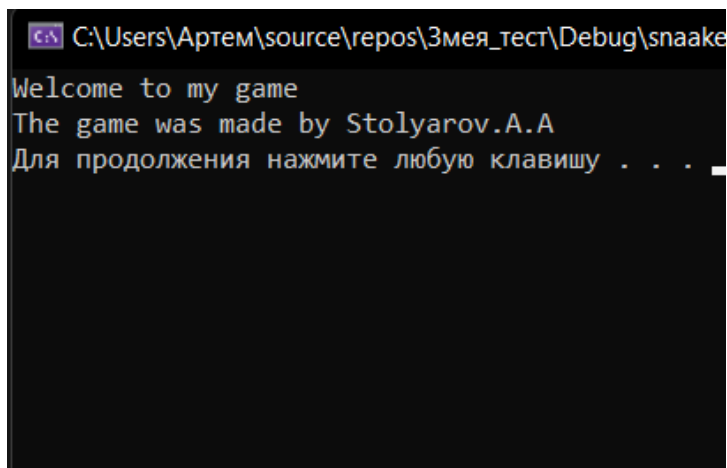


Рисунок 5 – Начало игры

После нажатия на любую клавишу у нас запускается сама игра.

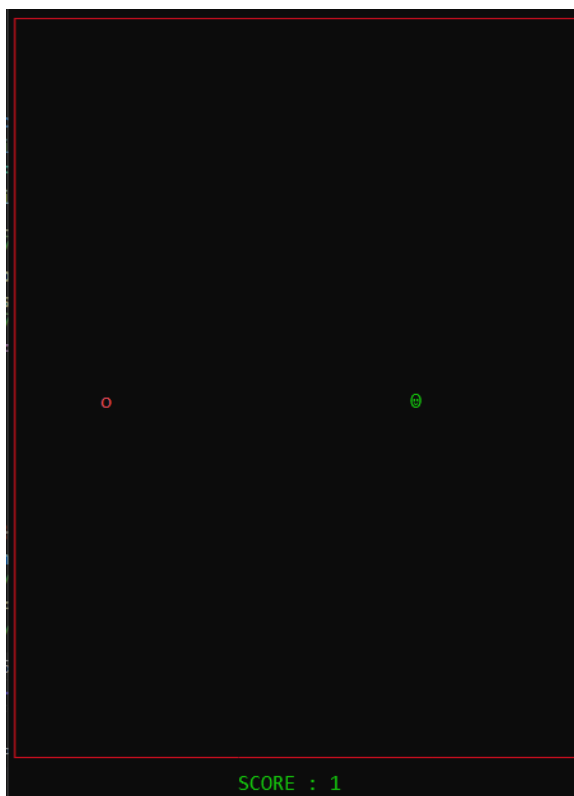


Рисунок 6 – Игровой процесс

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

←	↑	→	↓
Движение Влево	Движение Вверх	Движение Вправо	Движение Вниз

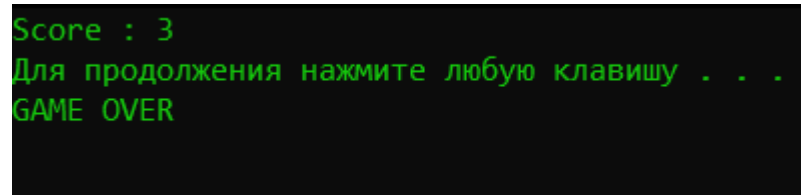
Рисунок 7 – Управление в игре

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



Рисунок 8 – Увеличение змейки и счета внизу от игрового поля

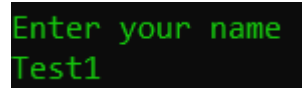
Если змейка врежется в себя или врежется в стену, ограничивающее игровое поле, то выводится сообщение о счете и об окончании игры.



```
Score : 3
Для продолжения нажмите любую клавишу . . .
GAME OVER
```

Рисунок 9 – Показ счета и завершение игры

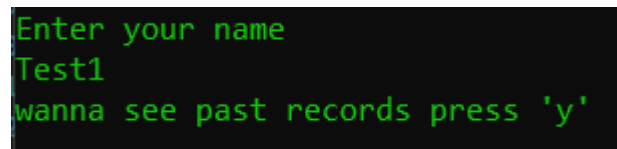
После этого, программа просит ввести свое имя, вводим:



```
Enter your name
Test1
```

Рисунок 10 – Ввод имени пользователя

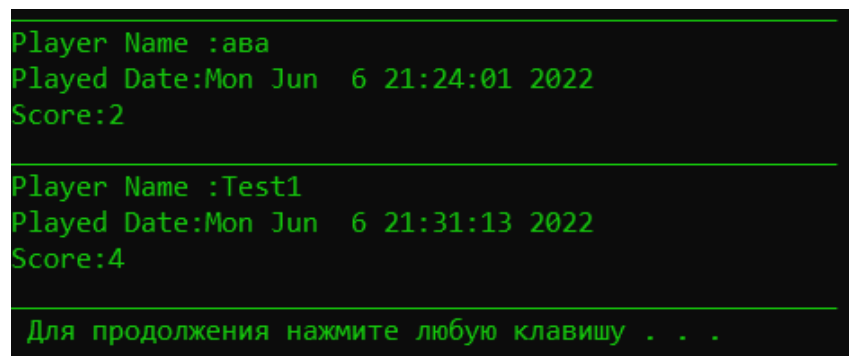
Далее, если мы хотим увидеть таблицу лидеров, нажимаем клавишу «Y».



```
Enter your name
Test1
wanna see past records press 'y'
```

Рисунок 11 – Предложение показа таблицы рекордов

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



```
Player Name :ава
Played Date:Mon Jun  6 21:24:01 2022
Score:2

Player Name :Test1
Played Date:Mon Jun  6 21:31:13 2022
Score:4

Для продолжения нажмите любую клавишу . . .
```

Рисунок 12 – Таблица рекордов

Заключение

При выполнении данной курсовой работы были улучшены базовые навыки программирования на языках C и Assembler.

Изучены основные возможности среды программирования Microsoft Visual Studio 2022. Были освоены способы обработки событий с помощью клавиатуры. Получены навыки отладки и тестирования программ. Были освоены приемы создания и подключения модулей программы. Получены навыки работы со сложными типами данных.

В результате была разработана игра «Змейка», с приятным оформлением и простым для использования интерфейсом.

В дальнейшем игру можно улучшить, добавив большее оформление игрового поля а также звуковое сопровождение. Улучшить визуальную составляющую игры, добавить более информативное меню, и оптимизировать код для более быстрого выполнения операций.

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

Файл Header.h

```
#pragma once

#define _CRT_SECURE_NO_WARNINGS

#include <iostream>

#include <windows.h>

#include <ctime>

#include <conio.h>
```

Файл Funk.h

```
#pragma once
```

```
void GotoXY(int x, int y);
```

Файл Funk.spp

```
#pragma once
```

```
#include "Header.h"
```

```
int x;
```

```
int y;
```

```
void GotoXY(int x, int y)
```

```
{
```

```
    HANDLE a;
```

```
    COORD b;
```

```
    fflush(stdout);
```

```
    b.X = x;
```

```
    b.Y = y;
```

```
    a = GetStdHandle(STD_OUTPUT_HANDLE);
```

```
    SetConsoleCursorPosition(a, b);
```

```
}
```

Файл Sourse.spp

```
#include "Header.h"
```

```
#include "Funk.h"
```

```
int main()
```

```
{
```

```
    srand(time(0)); // запуск генератора случайных чисел
```

```
void Ni_higgers(); {
```

```

char word_1[] = "Welcome to my game\n" "The game was made by Stolyarov.A.A\n";
__asm {
lea eax, word_1;
push eax;
call printf;
add esp, 4;
}
system("pause");

}

```

```

rand(); // холостой ход генератора случайный чисел
system("mode con cols=51 lines=31"); // установка размеров окна консоли
MoveWindow(GetConsoleWindow(), 50, 50, 2000, 2000, true); // установка стартовой позиции
окна консоли (50 и 50 - это пиксели
// относительно верхнего левого угла монитора
const int width = 50, height = 30; // размеры поля, по которому бежит змейка
const int max_length = 50; // установка максимальной длины "змейки"
int array_X[max_length]; // массив, хранящий абсциссы звеньев "змейки"
int array_Y[max_length]; // массив, хранящий ординаты звеньев "змейки"

int length = 1; // переменная длины "змейки"
array_X[0] = width / 2; // установка стартовой абсциссы "змейки"
array_Y[0] = height / 2; // установка стартовой ординаты "змейки"
int dx = 1, dy = 0; // создание смещений по осям для движения "змейки"
int X_apple; // абсцисса "яблока"
int Y_apple; // ордината "яблока"
do // цикл ставит координаты яблока случайным образом - но чтобы они не совпадали со
"змейкой"
{

```

```

X_apple = rand() % (width - 2) + 1;
Y_apple = rand() % (height - 2) + 1;
} while (X_apple != array_X[length - 1] && Y_apple != array_Y[length - 1]);

int sleep_time = 200; // переменная частоты кадров

char snake = '*'; // символ для отображения тела "змейки"
char apple = 'o'; // символ для отображения "яблока"
char head = 1; // символ для отображения головы "змейки"
COORD c; // объект для хранения координат
HANDLE h = GetStdHandle(STD_OUTPUT_HANDLE); // создание хендла потока вывода
CONSOLE_CURSOR_INFO cci = { sizeof(cci), false }; // создание параметров на отображение курсора
SetConsoleCursorInfo(h, &cci); //связывание параметров и хендла

SetConsoleTextAttribute(h, 4); // установка цвета, которым рисуется рамка поля
for (int y = 0; y < height; y++) // стандартный двойной цикл на отрисовку рамки
{
    for (int x = 0; x < width; x++)
    {
        char s;
        if (x == 0 && y == 0) // в верхнем левом углу поля - символ соответствующего угла
            s = 218;
        else if (x == 0 && y == height - 1) // нижний левый угол
            s = 192;
        else if (y == 0 && x == width - 1) // верхний правый угол
            s = 191;
        else if (y == height - 1 && x == width - 1) // нижний правый угол
            s = 217;
        else if (y == 0 || y == height - 1) // верхняя и нижняя граница поля
            s = 196;
    }
}

```

```

else if (x == 0 || x == width - 1) // левая и правая граница поля
s = 179;

else s = ' '; // во всех остальных случаях должен быть просто пробел (означает пустую область
поля)

putchar(s); // выводим символ
}

cout << endl;
}

c.X = X_apple; // связываем объект координат с позициями "яблока"
c.Y = Y_apple;
SetConsoleCursorPosition(h, c); // отправляем курсор на позицию "яблока"
SetConsoleTextAttribute(h, 12); // устанавливаем красный цвет для отрисовки "яблока"
putchar(apple); // отображаем символ "яблока"

c.X = array_X[0]; // связываем объект координат со стартовой позицией "змейки"
c.Y = array_Y[0];
SetConsoleCursorPosition(h, c); // отправляем курсор на позицию головы "змейки"
SetConsoleTextAttribute(h, 10); // устанавливаем зеленый цвет для отрисовки "змейки"
putchar(head); // отображаем символ головы "змейки"

bool flag = true; // переменная для управления ходом цикла
do // собственно цикл игры
{
int Score(int length);
{
int score;
GotoXY(20, 30);
score = length;
printf("SCORE : %d", (length));

```

```
}
```

```
Sleep(sleep_time); // задержка потока программы на заданный ранее интервал
```

```
if (_kbhit()) // проверяем, была ли нажата какая-либо клавиша и запускаем её обработку в случае ИСТИНЫ
```

```
{
```

```
int k = _getch(); // считываем код клавиши из буфера
```

```
if (k == 0 || k == 224) // если первый код - вспомогательный, считываем второй код
```

```
k = _getch();
```

```
switch (k) // пропускаем код нажатой клавиши внутрь оператора выбора
```

```
{
```

```
case 80: // если была нажата клавиша вниз
```

```
dy = 1; // то приращение по оси ординат будет положительным
```

```
dx = 0; // по оси абсцисс приращение нулевое
```

```
break;
```

```
case 72: // если вверх
```

```
dy = -1; // аналогично согласно геометрической логике
```

```
dx = 0;
```

```
break;
```

```
case 75: // если влево
```

```
dy = 0;
```

```
dx = -1;
```

```
break;
```

```
case 77: // если вправо
```

```
dy = 0;
```

```
dx = 1;
```

```
break;
```

```
case 27: // если была нажата клавиша ESC
```

```
flag = false; // устанавливаем флажок на ЛОЖЬ, чтоб закончить показ движения
```

```
break;
```

```

}
}

int X = array_X[length - 1] + dx; // определяем значение абсциссы головы "змейки" после
смещения
int Y = array_Y[length - 1] + dy; // то же самое для ординаты
if (X == 0 || X == width - 1 || Y == 0 || Y == height - 1) // проверка на достижение границ поля
{
    flag = false; // пока что - просто установка управляющей переменной цикла
}
else if (X == X_apple && Y == Y_apple) // проверка на достижение "яблока"
{
    c.X = array_X[length - 1]; // установка в объект координат позиции головы "змейки"
    c.Y = array_Y[length - 1];
    SetConsoleCursorPosition(h, c); // установка курсора в эту позицию
    putchar(snake); // отображение символа тела "змейки"

    length++; // увеличение длины "змейки" (яблоко проглочено)
    c.X = array_X[length - 1] = X; // установка в массивы позиции нового звена "змейки"
    c.Y = array_Y[length - 1] = Y;
    SetConsoleCursorPosition(h, c); // установка туда курсора
    putchar(head); // и отображение там символа головы "змейки"

    if (length == max_length) // проверка, достигла ли длина "змейки" своего максимального
значения
    {
        break; // пока что - просто прерываем цикл
    }

    int i; // переменная для подсчета количества звеньев "змейки", не совпадающих с позицией
"яблока"
    do
    {

```

```

X_apple = rand() % (width - 2) + 1; // установка новых координат "яблока"
Y_apple = rand() % (height - 2) + 1;
i = 0; // обнуление числа несовпадающих координат
for (; i < length; i++) // запуск цикла на сверку совпадений
if (X_apple == array_X[i] && Y_apple == array_Y[i]) // если совпадение найдено
break; // то прерываем цикл for
} while (i < length); // поиск новых координат продолжается, пока число несовпадающих
координат меньше длины "змейки"

c.X = X_apple; // установка в объект координат новой корректной позиции "яблока"
c.Y = Y_apple;
SetConsoleCursorPosition(h, c); // отправка туда курсора
SetConsoleTextAttribute(h, 12); // установка цвета в красный
putchar(apple); // отображение символа "яблока"
SetConsoleTextAttribute(h, 10); // обратная установка цвета в зеленый - для дальнейшего
отображения "змейки"
}

else // случай, когда голова "змейки" оказалась на новой пустой позиции
{
int i = 1; // переменная на количество звеньев, не совпадающих с новой позицией - кроме
хвоста "змейки"
for (; i < length; i++)
if (X == array_X[i] && Y == array_Y[i]) // если совпадение найдено в цикле - прерываемся
break;
if (i < length) // если число несовпадающих звеньев меньше длины "змейки" - то прерываем
основной цикл игры
{
break;
}
else // а иначе запускаем обработку сдвига "змейки"
{
c.X = array_X[0]; // устанавливаем в объект координат позицию хвоста "змейки"

```

```

c.Y = array_Y[0];
SetConsoleCursorPosition(h, c); // двигаем туда курсор
putchar(' '); // и отображаем пробел (затирка хвоста)

if (length > 1) // если длина змейки больше
{
c.X = array_X[length - 1]; // устанавливаем в объект координат предыдущую позицию головы
"змейки"
c.Y = array_Y[length - 1];
SetConsoleCursorPosition(h, c); // двигаем туда курсор
putchar(snake); // выводим символ тела "змейки"
}

for (int i = 0; i < length - 1; i++) // запускаем цикл сдвига координат звеньев "змейки"
{
array_X[i] = array_X[i + 1]; // обрабатываем все звенья - кроме последнего
array_Y[i] = array_Y[i + 1];
}

c.X = array_X[length - 1] = X; // устанавливаем новую позицию головы "змейки"
c.Y = array_Y[length - 1] = Y;
SetConsoleCursorPosition(h, c); // двигаем туда курсора
putchar(head); // отображаем символ головы "змейки"
}
}

} while (flag); // выходим из цикла, если сброшена управляющая переменная
system("cls");// очищаем экран
int Score(); {
int score;

```

```

score = length;
printf("Score : %d\n", length++);
system("pause");
}

cout << "GAME OVER\n"; // выводим сообщение о конце игры
void record(); {
char pname[20], nplname[20], cha, c;
int i, j, px;
FILE* info;
info = fopen("record.txt", "a+");
_getch();
//system("cls");
printf("Enter your name\n");
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", px = length);//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);
}
system("pause");
}

```


Приложение Б

Скриншоты программы

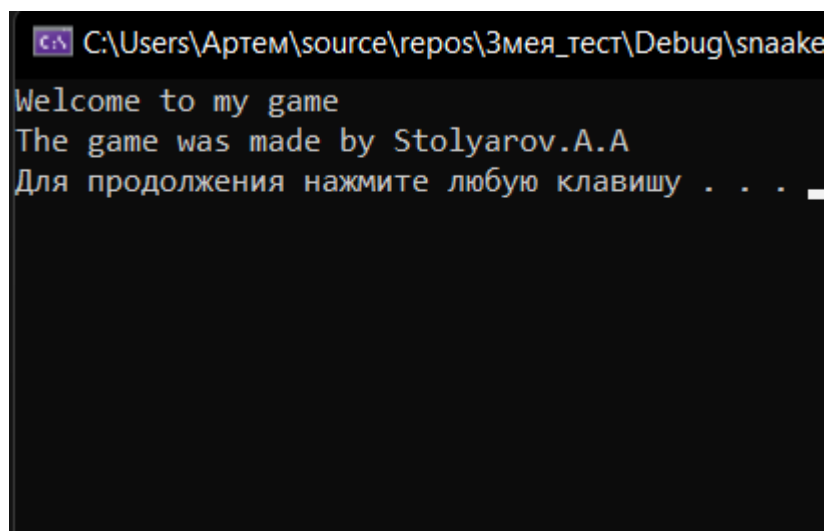


Рисунок 13 – Заставка

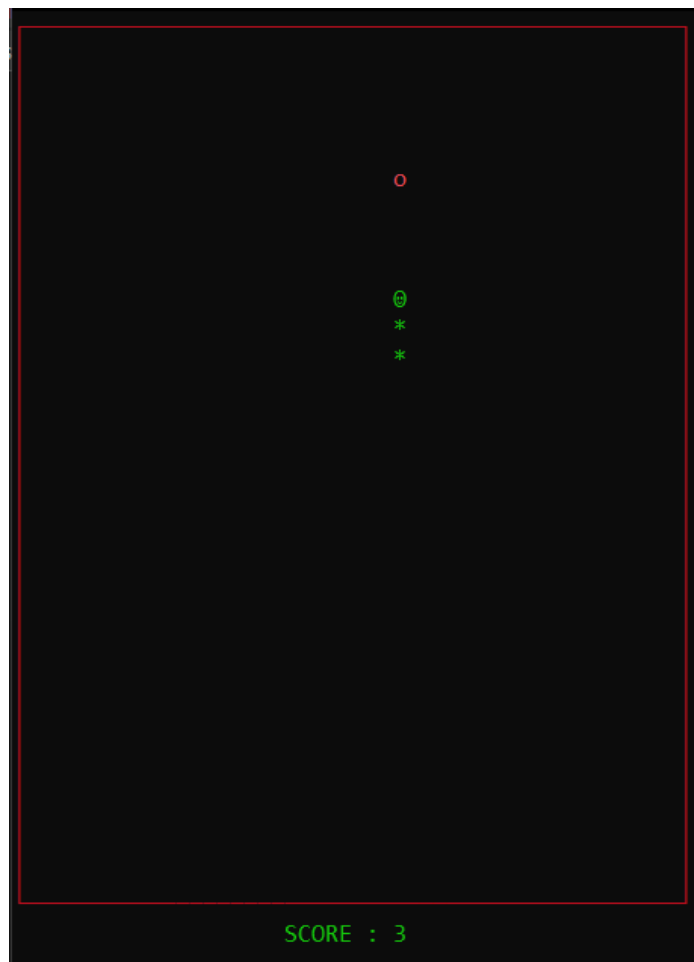


Рисунок 14 – Игровой процесс

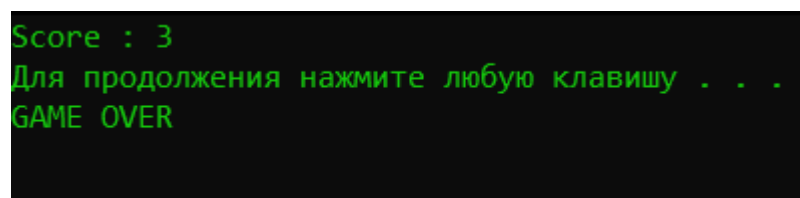


Рисунок 15 – Конец игры

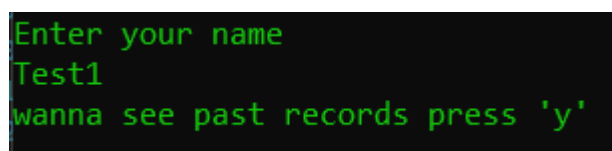


Рисунок 16 – Ввод имени

```

Player Name :ава
Played Date:Mon Jun  6 21:24:01 2022
Score:2

Player Name :Test1
Played Date:Mon Jun  6 21:31:13 2022
Score:4

Для продолжения нажмите любую клавишу . . .

```

Рисунок 17 – Вывод таблицы лидеров

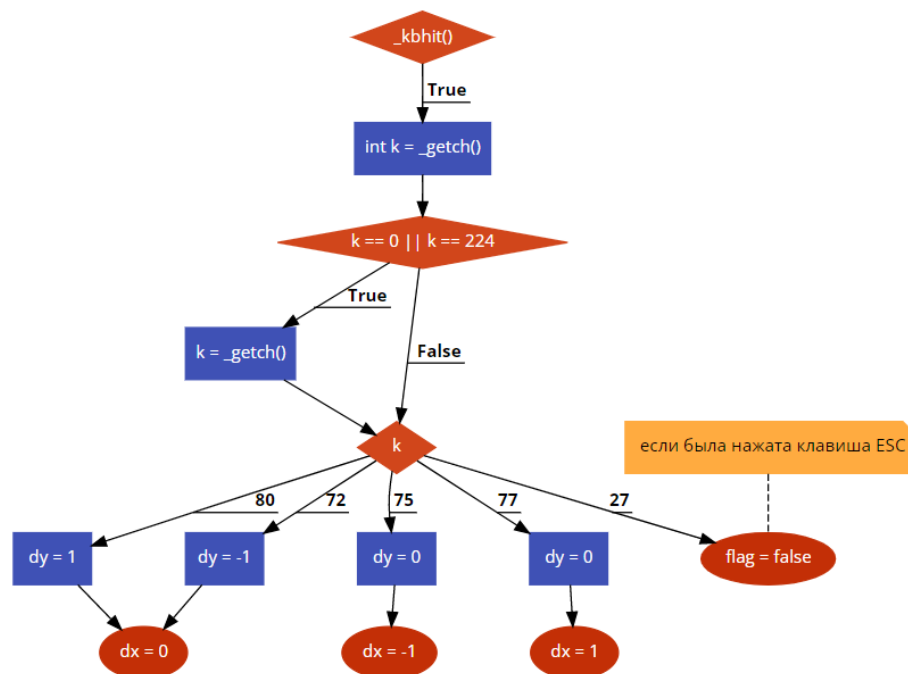


Рисунок 18 – Управление с клавиатуры

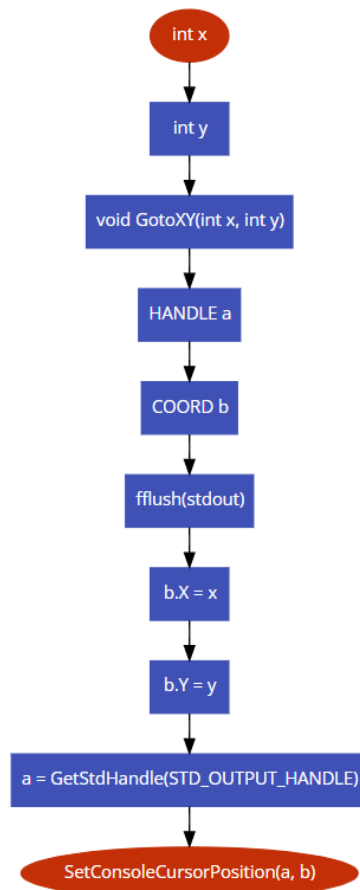


Рисунок 19 – Абсцисса X и абсцисса Y для вывода данных

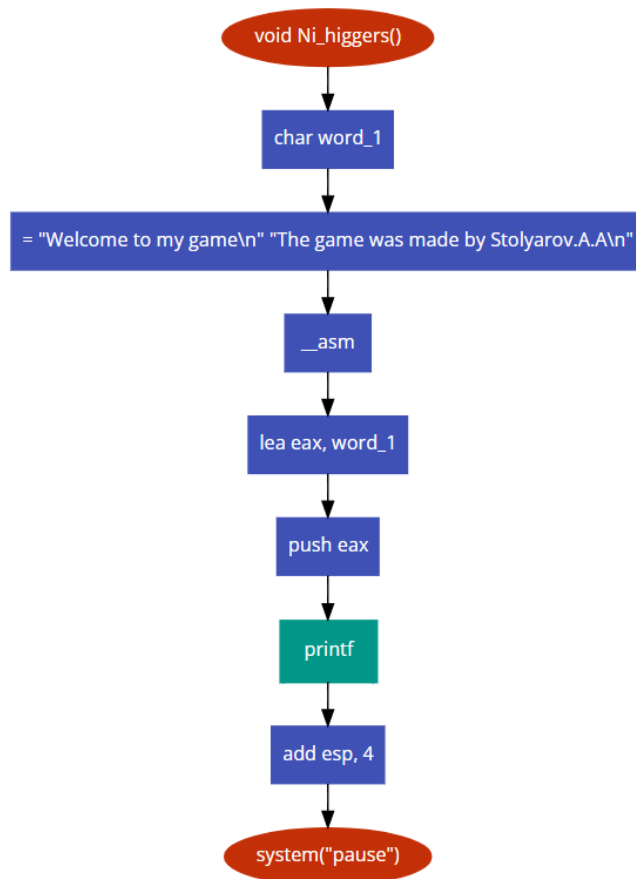


Рисунок 19 – Assembler вставка

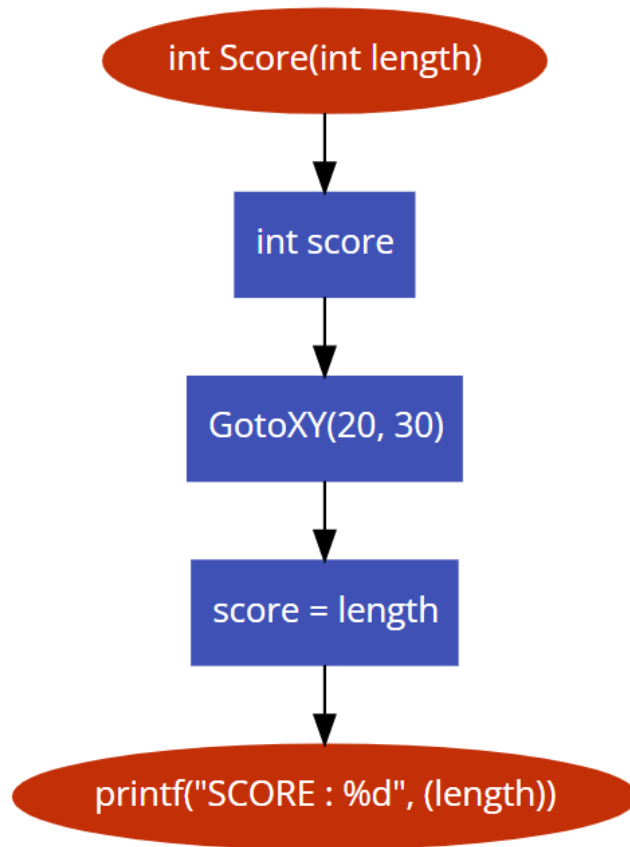


Рисунок 20 – Вывод счета