

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

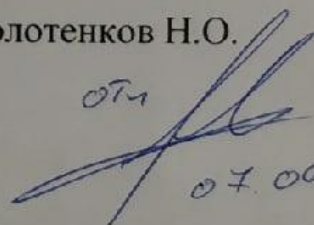
21ВВ2

Митрошин

Александр

Принял:

Голотенков Н.О.

от

07.06.22

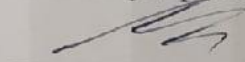
Пенза 2022

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 Митрохину Александру Дмитриевичу Группа 2113132

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

Исходные данные (технические требования) на проектирование

Приложение: "Змейка"
Функции: размер поля не менее 15 квадратов, управление игро-
виной, возможность сохранения имени пользователя с
результатами игры, возможность вывода последних 15 пользо-
вателей на экран из файла.

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 " 02.22

Дата защиты проекта " 07 " 06 22

Руководитель Гончаренков Н.О.

Задание получил " 22 " 02 2022г.

Студент Солдатов А.Д.

Содержание

Введение	5
1. Постановка задачи и обязательные требования	6
2. Выбор решения	7
3. Описание разработки программы	7
4. Отладка и тестирование.....	8
5. Описание программы.....	9
5.1 Функции	9
5.2 Разработка функций на языке Assembler.....	10
6. Руководство пользователя.....	11
Заключение.....	15
Список используемых источников	16
Приложение А. Иерархическая структура программы.....	17
Приложение В. Схема данных программы	18
Приложение С. Листинг программы	19
Приложение D. Схемы функций программы	28
Приложение Е. Схема взаимодействия программы.....	32

Введение

Компьютерные технологии уже давно укоренились в нашей жизни. Многие привычные для нас вещи были бы невозможны без программирования.

Язык программирования Си - это универсальный язык с богатым набором операторов и компактным способом записи выражений. Благодаря гибкости, выразительности и компактности своих конструкций. Си завоевал наибольшую популярность в среде профессиональных программистов и широко используется при разработке системных и прикладных программ.

В данной работе продемонстрированы базовые возможности языка СИ вкупе с ассемблерными вставками. Разработка велась в среде Microsoft Visual Studio 2010.

1. Постановка задачи и обязательные требования

Необходимо разработать программу «Игра змейка», которая должна отвечать следующим требованиям:

1. Многомодульность. Программа должна быть поделена на логические модули. Это упростит дальнейшее совершенствование программы, а также позволит локализовать ошибки.
2. Использование сложных типов данных: массивов, структур, объектов и т.д. Для разработки данной игры необходимо подробно изучить принципы работы с массивами, структурами и объектами для создания игрового поля и реализации графики.
3. Режим работы видеосистемы: текстовый/графический. Необходимо определиться с типом управления, видом интерфейса и изучить приёмы реализации графики.
4. Устройство ввода информации: клавиатура/мышь. Необходимо определиться с механикой взаимодействия игрока с игрой и её интерфейсом.
5. Пользовательский интерфейс должен быть построен на основе меню или панели инструментов. Необходимо реализовать пользовательское меню, с заставкой.
6. Операционная система: MS Windows. При разработке использовать среду MS Windows с системой программирования MS Visual Studio.
7. Язык программирования: Си и Ассемблер, а также некоторые функции из языка C++.

2. Выбор решения

Приложение «zmeyka» состоит из двух модулей:

1. zmeyka.cpp
2. zm.cpp

Файл «zmeyka.cpp» является основным модулем данной курсовой работы, которая контролирует основные функции и вызов других функций из других файлов. Он связан заголовочным файлом «Zmeyka.h».

Файл «zm.cpp» используется для вывода заставки на консоль.

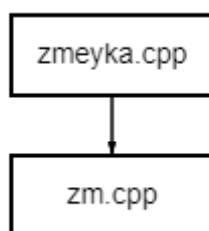


Рисунок 1 — Структура модулей программы

3. Описание разработки программы

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. При разработке языка Си был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. В языке Си предусмотрено много операций, непосредственно исполняемых большинством микропроцессоров; в то же время он дает программисту возможность выражать свои алгоритмы и данные наиболее подходящими средствами. Язык Си обеспечивает возможности структурирования данных. Он позволяет

разрабатывать большие, но структурированные программы, предоставляя возможность отдельной разработки подпрограмм.

Целью создателей языка Си была разработка мобильного языка, который можно было бы использовать для разработки системного программного обеспечения. Программы, написанные на языке высокого уровня, мобильны, поскольку язык должен быть одним и тем же независимо от того, на каком компьютере и в какой операционной системе он используется. Дополнительное свойство языка Си, называемое условной компиляцией, позволяет программисту изолировать машинно-зависимые операторы и контролировать их компиляцию в другой среде. Это дополнительно повышает мобильность программного обеспечения, написанного на языке Си.

В качестве среды программирования был выбран программный продукт VisualStudio 2010.

Интерфейс игры будет представлен в виде консоли. Ход совершается нажатием любой из стрелочных клавиш. Результат будет представлен на экране, а также сохранён в документе «result.txt», появляющемся в корневой папке после завершения игры.

Реализация программы началась с определения необходимых функций для отображения карты, тела змейки и фрукта и их начального положения.

Основная программа zmeuka.cpp выполняется функцией main. По завершению игры имя игрока и его счёт заносятся в файл.

4. Отладка и тестирование

В качестве среды разработки была выбрана программа Visual Studio 2010. Программа предоставляет все средства, необходимые

при разработке и отладке модулей и программ. В ходе отладки применялись такие возможности, как точка остановки, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти.

Отладчик — это узкоспециализированное средство разработки, которое присоединяется к работающему приложению и позволяет проверять код. В документации по отладке для Visual Studio именно это обычно подразумевается под отладкой. Отладчик является важнейшим инструментом для поиска и устранения ошибок в приложениях.

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

5. Описание программы

5.1 Функции

Приложение zmeuka.cpp является основным модулем программы. После запуска программы выводится заставка. После программа ожидает ввода имени пользователя, затем отрисовывается игровое поле, голова змейки и фрукт.

Setup - отвечает за положение змейки и фрукта. Задаётся значение для координатных переменных. Задаются координаты для фрукта таким образом, чтобы они каждый раз принимали случайное значение. И добавляется переменная счётчика.

Draw - отрисовка карты, змейки и фрукта. Сначала очищается экран консоли, затем заполняется верхняя часть поля. После этого задаём отрисовку для правой и левой стенок. Добавляем символы отображения головы змейки и её тела и фрукта. Заполняем нижнюю часть поля и под ним отображаем счётчик.

Input - отвечает за перемещение змейки. Определяем каждому нажатию кнопки значение направления движения змейки

Logic - отвечает за логику всей игры. Делаем так, чтобы движение хвоста каждый раз «обновлялось» и соответственно принимало новые значения координат. Делаем так, чтобы змейка могла выходить за поле и появляться в диаметрально противоположном месте. Если случается так, что координаты хвоста совпали с координатой головы змейки, то есть змейка съела саму себя, то игра заканчивается. Если координаты головы змейки совпали с координатами фрукта, то есть змейка съедает фрукт, то длина её хвоста увеличивается на один символ, а к значению счётчика добавляется «10».

resultat - отвечает за запись в файл имени игрока и закреплённый за ним счёт.

В файле «zm.cpp» находится функция start_screen , которая отображает заставку в консоле.

5.2 Разработка функций на языке Assembler

Как известно, функции, реализованные на языке ассемблера, занимают меньше места и работают быстрее. В качестве разрабатываемых функций были выбраны функции для обнуления поля, хода игрока и проверки.

Описание функций выполнено на языке C, а вычисления (частично, в некоторых случаях приходилось использовать язык СИ) представляют собой ассемблерную вставку. Для реализации необходимо перед такой вставкой прописать ключевое слово «__asm», а сам код расположить в фигурных скобках.

Были использованы следующие ассемблерные инструкции:

- команда `mov` используется для копирования значения из источника в приемник;
- команда `idiv` производит целочисленное деление с остатком знаковых целочисленных операндов;
- команда `dec` уменьшает число на единицу;

```
33  _asm {  
34      mov eax, width  
35      mov ebx, 2  
36      cdq  
37      idiv ebx  
38      dec eax  
39      mov x, eax  
40      mov eax, high  
41      cdq  
42      idiv ebx  
43      dec eax  
44      mov y, eax  
45  }  
46  //x = width / 2 - 1;  
47  //y = high / 2 - 1;
```

Рисунок 2 — Ассемблерная вставка

Переменную `width` делим на 2 и вычитаем из неё единицу и записываем, получившийся результат в переменную, отвечающую за координату `x`. Аналогичные действия производятся для переменной, отвечающей за координату `y`.

6. Руководство пользователя

Программа `zmeuka` предназначена для игры в змейку. Она имеет понятный интерфейс и интуитивно понятное управление.

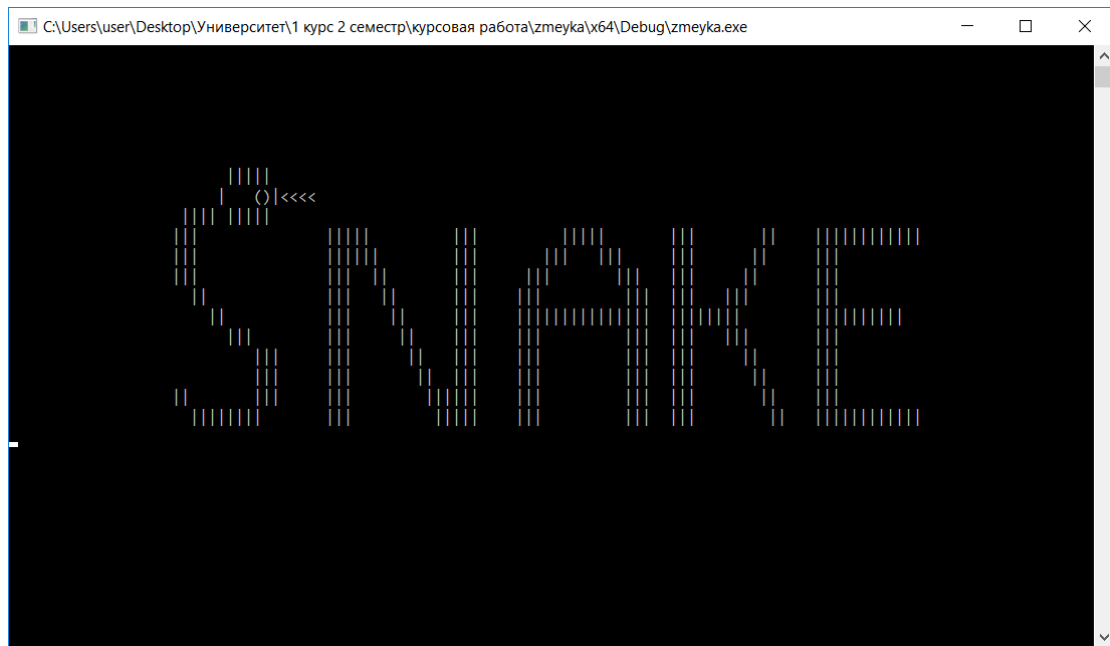


Рисунок 3.1 — Заставка

Вводится имя игрока и нажимается Enter:

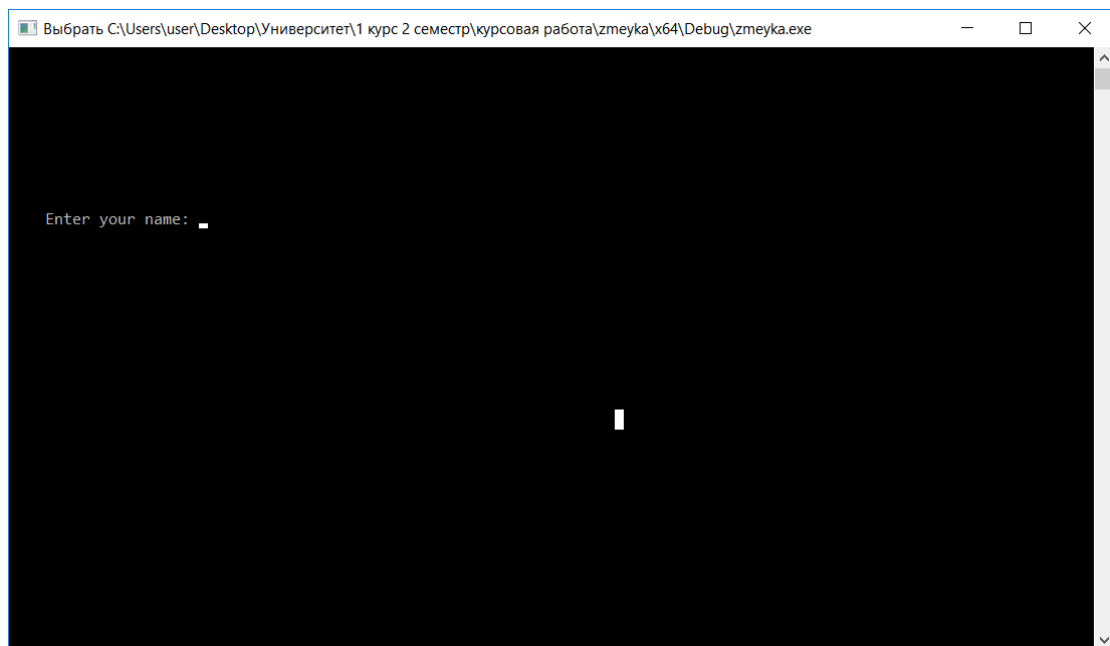


Рисунок 3.2 — Ввод имени пользователя

Отображается поле игры, голова змейки, случайным образом расположен фрукт и под полем находится счётчик:

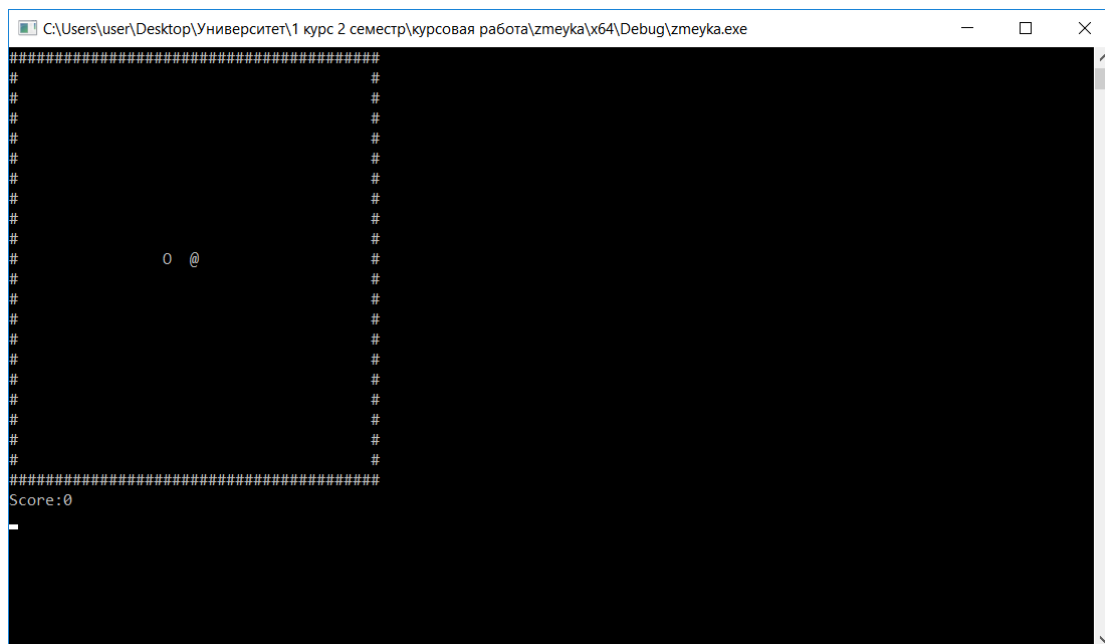


Рисунок 3.3 — Игровое поле

Каждый раз, когда змейка съедает фрукт, длинна её тела увеличивается на один символ и добавляется 10 баллов к общему счёту:

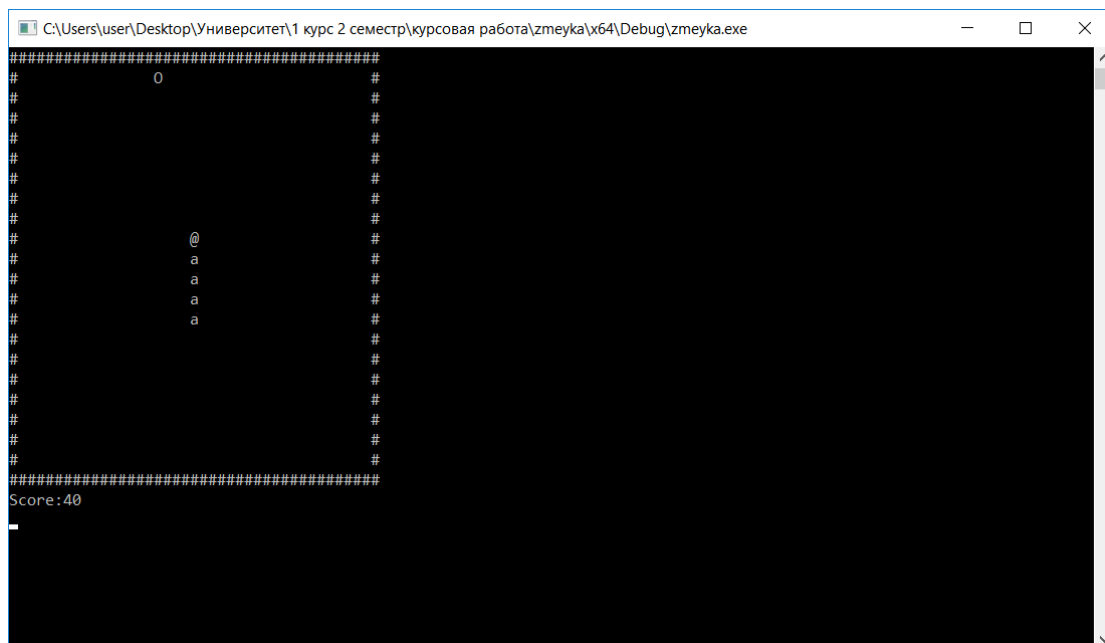


Рисунок 3.4 — Ход игры

Когда змейка съедает/врезается в себя, то игра заканчивается:

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Были освоены способы обработки событий от клавиатуры. Усвоены принципы реализации интерфейса. Получены базовые навыки программирования на языке С и Ассемблер. Изучены возможности среды программирования VisualStudio 2010. Получены навыки отладки и тестирования программ.

В рамках выполнения курсовой работы была написана программа (игра) змейка. Программа не требует большого количества системных ресурсов, обеспечивает быструю обработку данных и мгновенный показ результата пользователю.

Программу можно улучшить, оптимизировав код для более быстрого выполнения функций.

Список используемых источников

1. <https://opita.net/node/876>
2. <https://learnc.info/c/functions.html>
3. <http://natalia.appmat.ru/c&c++/assembler.html>
4. <https://habr.com/ru/post/423077/>

Приложение А. Иерархическая структура программы

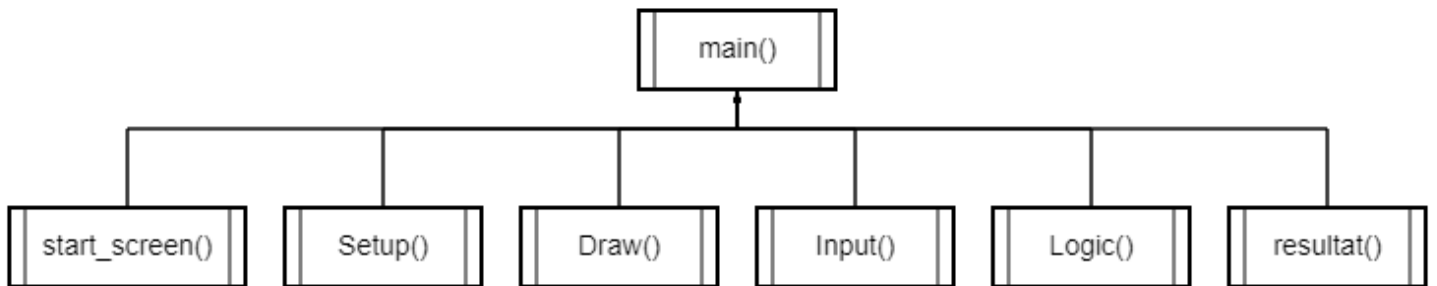


Рисунок 4 — Иерархическая структура программы

Приложение В. Схема данных программы

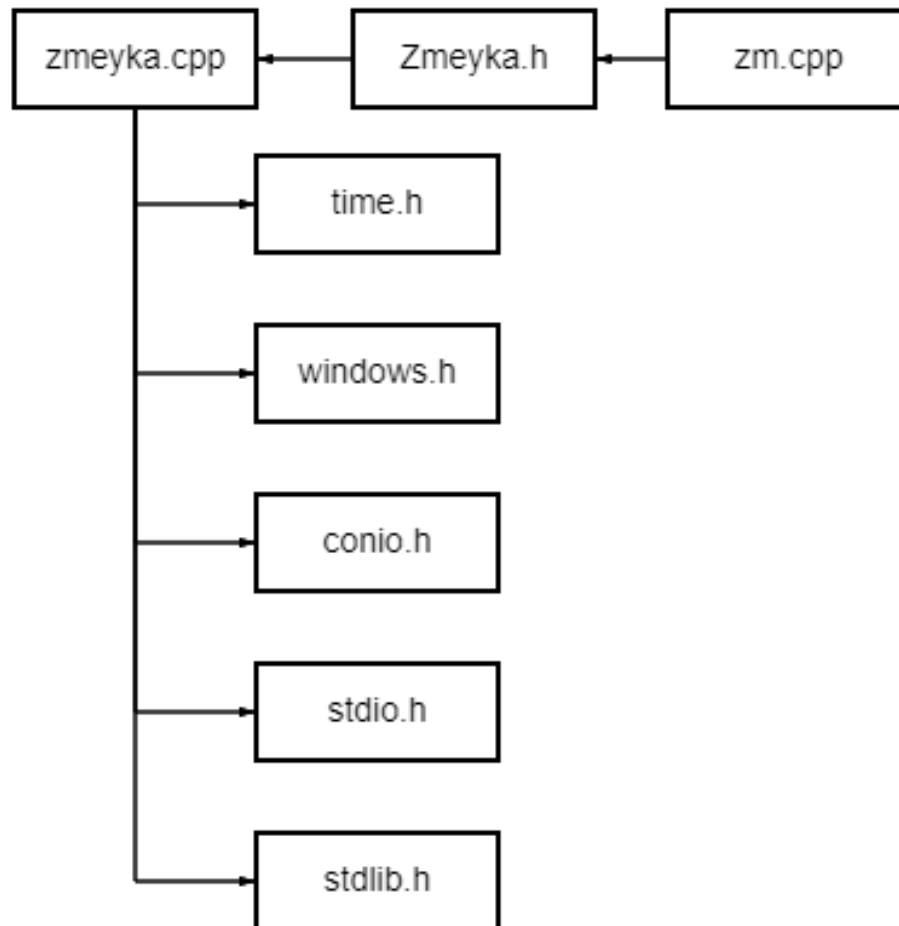


Рисунок 5 — Схема данных программы

Приложение С. Листинг программы

```
#include <iostream>//библиотека для ввода/вывода//
#include <time.h>
#include <windows.h>
#include <ctime>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include <vector>
#include "Zastavka.h"
using namespace std;
bool GameOver;
const int width = 40;// Ширина поля//
const int high = 20;// Длинна поля//
int x, y, fruitX, fruitY, score;
int tailX[100], tailY[100];
int ntail;
int s = 10000;
enum eDirection { STOP = 0,LEFT, RIGHT, UP, DOWN };
//присвоение параметрам целые значения//
eDirection dir;

void Setup() {// Отвечает за положение змейки и фрукта//
    srand(time(0));
    rand();
    GameOver = false;
```

```

dir = STOP;
_asm {
    mov eax, width
    mov ebx, 2
    cdq
    idiv ebx
    dec eax
    mov x, eax
    mov eax, high
    cdq
    idiv ebx
    dec eax
    mov y, eax
}

//x = width / 2 - 1;
//y = high / 2 -1;
fruitX = rand() % width;
fruitY = rand() % high;
score = 0; // Счётчик//
}

void Draw() {// Отрисовка карты, змейки и фрукта//
    system("cls");// Очищает экран консоли//
    for (int i = 0; i < width + 1; i++)// Заполнение верхнюю часть
поля//
        printf( "#");
    printf(" \n");// Переход на вторую строку консоли//
}

```

```

        for (int i = 0; i < high; i++) {
            for (int j = 0; j < width; j++) {
                if (j == 0 || j == width - 1)
                    cout << "#"; // Заполнение левой и правой
стенки поля //
                if (i == y && j == x) // Задаём символ отображения
змейки //
                    cout << "@";
                else if (i == fruitY && j == fruitX)
                    cout << "O"; // Задаём символ отображения
фрукта //
                else {
                    bool print = false;
                    for (int k = 0; k < ntail; k++) {
                        if (tailX[k] == j && tailY[k] == i) {
                            print = true;
                            cout << "a";
                        }
                    }
                    if (!print)
                        cout << " ";
                }
            }
        }
        cout << endl;
    }

```

```
        for (int i = 0; i < width + 1; i++)// Заполнение нижней части  
поля//
```

```
        cout << "#";
```

```
    cout << endl;
```

```
    cout << "Score:" << score << endl;
```

```
}
```

```
void Input() {// Отвечает за перемещение змейки//
```

```
    Sleep(100);
```

```
    if (_kbhit()) {
```

```
        switch (_getch())
```

```
        {
```

```
        case 75:
```

```
            dir = LEFT;
```

```
            break;
```

```
        case 77:
```

```
            dir = RIGHT;
```

```
            break;
```

```
        case 72:
```

```
            dir = UP;
```

```
            break;
```

```
        case 80:
```

```
            dir = DOWN;
```

```
            break;
```

```
        case 27:
```

```
            GameOver = true;
```

```
            break;
```

```
        }
```

```
}  
}
```

```
void Logic() {// Отвечает за логику игры//
```

```
    int preX = tailX[0];
```

```
    int preY = tailY[0];
```

```
    int pre2X, pre2Y;
```

```
    tailX[0] = x;
```

```
    tailY[0] = y;
```

```
    for (int i = 1; i < ntail; i++) {
```

```
        pre2X = tailX[i];
```

```
        pre2Y = tailY[i];
```

```
        tailX[i] = preX;
```

```
        tailY[i] = preY;
```

```
        preX = pre2X;
```

```
        preY = pre2Y;
```

```
    }
```

```
    switch (dir) {
```

```
    case LEFT:
```

```
        x--;
```

```
        break;
```

```
    case RIGHT:
```

```
        x++;
```

```
        break;
```

```
    case UP:
```

```
        y--;
```

```
        break;
```

case DOWN:

 y++;

 break;

}

if (x >= width - 1)

 x = 0;

else if (x < 0)

 x = width - 2;

if (y >= high)

 y = 0;

else if (y < 0)

 y = high - 2;

for (int i = 0; i < ntail; i++) {

 if (tailX[i] == x && tailY[i] == y)

 GameOver = true;

}

if (x == fruitX && y == fruitY) {

 score += 10;

 fruitX = rand() % width;

 fruitY = rand() % high;

 ntail++;

}

}

```
void resultat(string str, int score)
{
    ofstream res("result.txt", ios::ate | ios::out | ios::app);
    res << str << " " << score << endl;
    res.close();
}
```

```
int main() {
    start_screen();
    string str;
    printf(" \n \n\n\n\n\n\n \n  Enter your name: ");
    cin >> str;
    cout << endl;
    Setup();
    while (!GameOver) {
        Draw();
        Input();
        Logic();
    }
    resultat(str, score);
    return 0;
}
```

zm.cpp

```
#include <iostream> //библиотека для ввода/вывода//
```

```
#include <time.h>
```

```
#include <windows.h>
```

```
#include <ctime>
```

```
#include <conio.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <fstream>
```

```
#include <vector>
```

```
void start_screen() {
```

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

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

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

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

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

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

```
    printf("          ||||                                     \n");
```

```
    printf("          |  O|<<<<                                     \n");
```

```
    printf("          ||| ||||                                     \n");
```

```
    printf("          ||          ||||          ||          ||||          ||          ||          \n");
```

```
    printf("          ||          ||||          ||          ||  ||          ||          ||          \n");
```

```
    printf("          ||          ||  ||          ||          ||          ||          ||          \n");
```

```
    printf("          ||          ||  ||          ||          ||          ||          ||          \n");
```

```
    printf("          ||          ||  ||          |||||          |||||          \n");
```

```

printf("          |||      |||  ||  |||  |||      ||| |||  |||      |||\n");
printf("          |||      |||  ||  |||  |||      ||| |||  ||      |||\n");
printf("          |||      |||      ||  |||  |||      ||| |||      ||      |||\n");
printf("      ||      |||      |||      |||||  |||      ||| |||      ||      |||\n");
printf("      ||||| |||      |||||  |||      ||| |||      ||      |||
|||||||)\n");
Sleep(2000);
system("cls");
}

```

Приложение D. Схемы функций программы

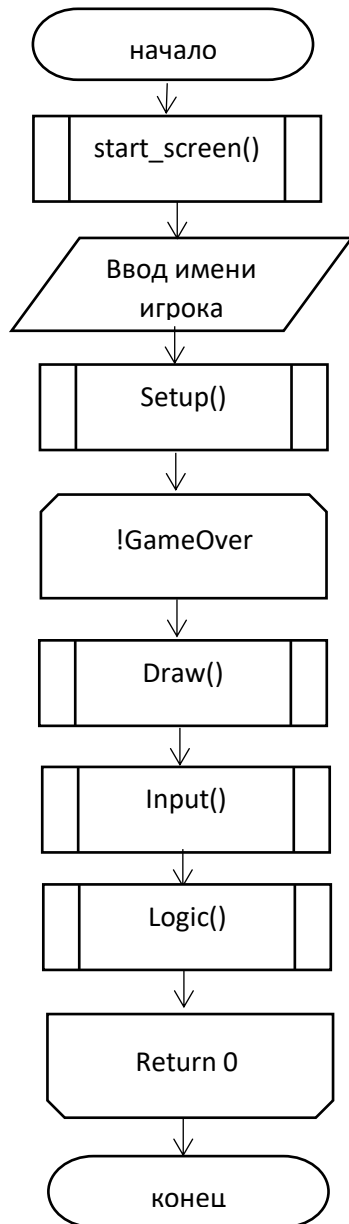


Рисунок 6.1 — Функция main



Рисунок 6.2 — Функция Setup

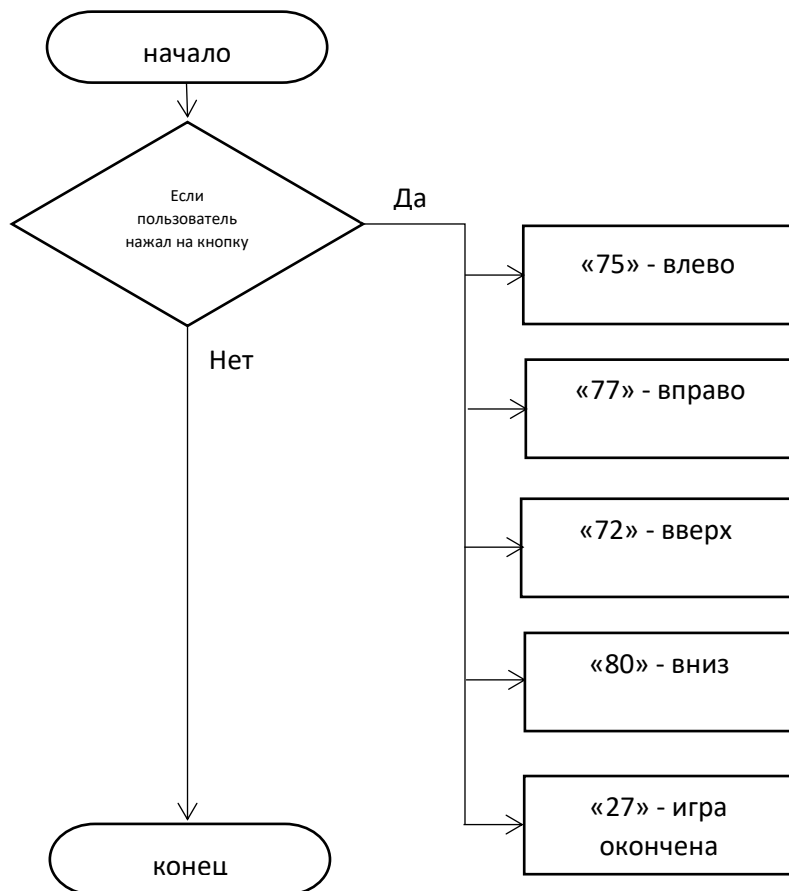


Рисунок 6.3 — Функция Input



Рисунок 6.4 — Функция Draw

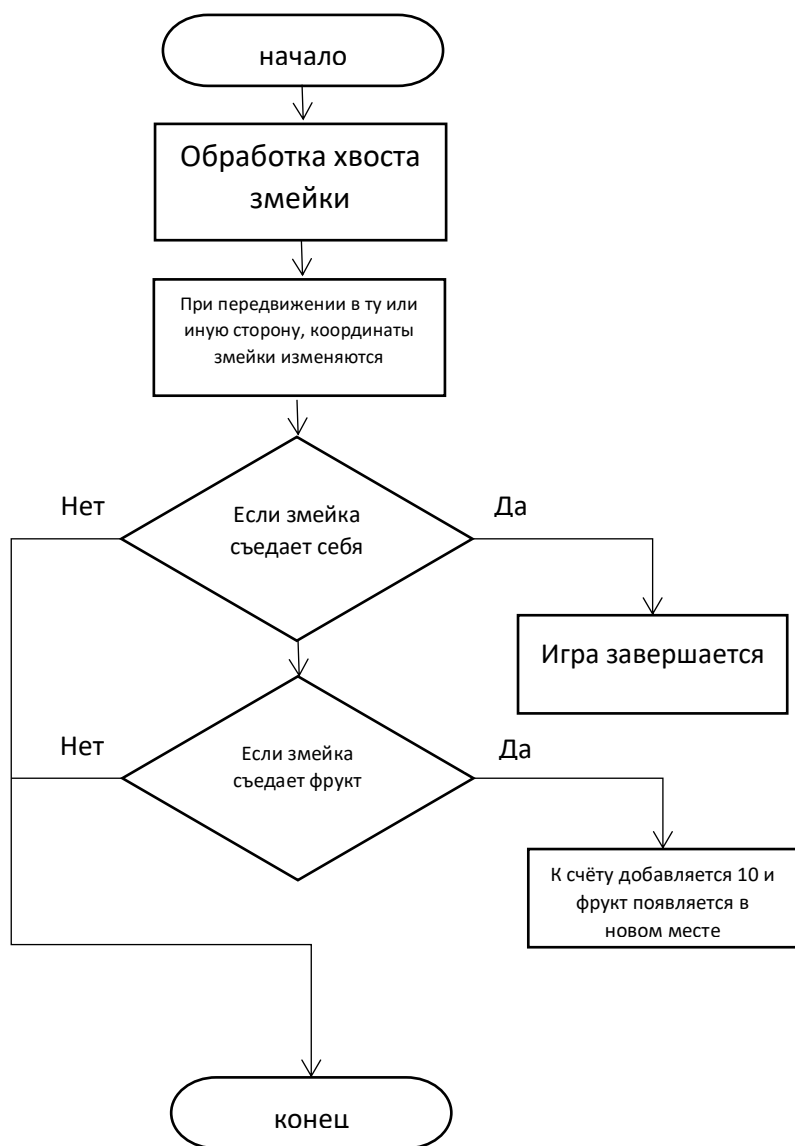


Рисунок 6.5 — Функция Logic

Приложение Е. Схема взаимодействия программы

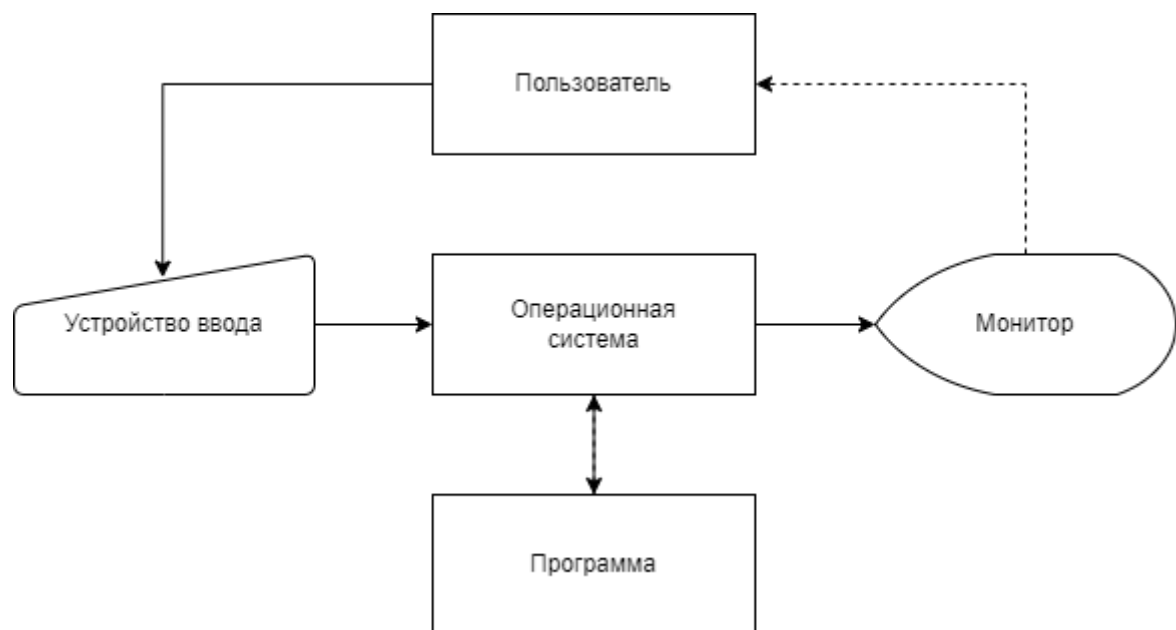


Рисунок 7 — Схема взаимодействия программы