

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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Игра “Змейка”»

Выполнил:

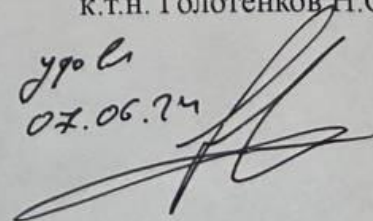
студент группы 23BBB2

Федоров Б.М.

Принял:

к.т.н. Голотенков Н.О.

ур. 6
07.06.24



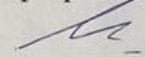
Пенза, 2024

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 Федорову Борису Максимовичу Группа 23ВВА2

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

Исходные данные (технические требования) на проектирование

Наименование программы: *Игра "Змейка"*

Функция программы: *управление движением змейки с помощью клавиатуры, система набора очков, рост змейки, изменение цветовой палитры*

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 "1" 03.2024

Дата защиты проекта "07" 06.24

Руководитель Голоменко Н.О.

Задание получил "1" марта 2024г.

Студент Федоров Б.М.

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	12
Заключение	16
Список используемых источников.....	17
Приложение А. Иерархическая структура программы.....	18
Приложение В. Схема данных программы	19
Приложение С. Листинг программы.....	20
Приложение D. Схемы функций программы	30
Приложение Е. Схема взаимодействия программы	36

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.

С развитием цифровых технологий компьютеры все больше вливаются в жизнь человека. Если раньше ЭВМ использовались исключительно для сложных математических вычислений, то сегодня сфера их применения существенно расширилась. Компьютерные игры - одно из наиболее массовых применений электронных вычислительных машин.

Развитие игровой индустрии шло стремительным темпом, и особенно пользовалось популярностью у подростков. Первые игры отличались простотой интерфейса и логики, но со временем они становились все сложнее и сложнее, над их созданием работал уже не один человек, а целая команда разработчиков.

Современные игры требуют достаточно большой производительности от компьютера, и не каждая офисная машина в силах воспроизводить их. Однако для отдыха от монотонной работы зачастую достаточно простой, не требовательной к технике, игры. Одной из таких является игра «Змейка».

Впервые эта игра возникла еще в 1977 году, но и по сей день, она пользуется популярностью, особенно на мобильных платформах. Игрок управляет длинным, тонким существом, напоминающим змею, которое ползает по плоскости (как правило, ограниченной стенками), собирая еду (или другие предметы), избегая столкновения с собственным хвостом и краями игрового поля.

В данной работе продемонстрированы базовые возможности языка СИ в

совокупности с ассемблерными вставками. Разработка велась в среде Microsoft Visual Studio 2022.

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.

2. Выбор решения

Приложение «Snake» состоит из двух модулей:

1. Snake.cpp
2. stfn.cpp

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

Файл «stfn.cpp» используется для вывода заставки и окончания игры в консоли.

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.

В качестве среды программирования был выбран программный продукт VisualStudio 2022.

Интерфейс игры будет представлен в виде консоли. Ход совершается нажатием любой из клавиш W, A, S или D. Результат будет представлен на экране, а также сохранён в документе «score.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.

Программа Snake.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 2022. 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

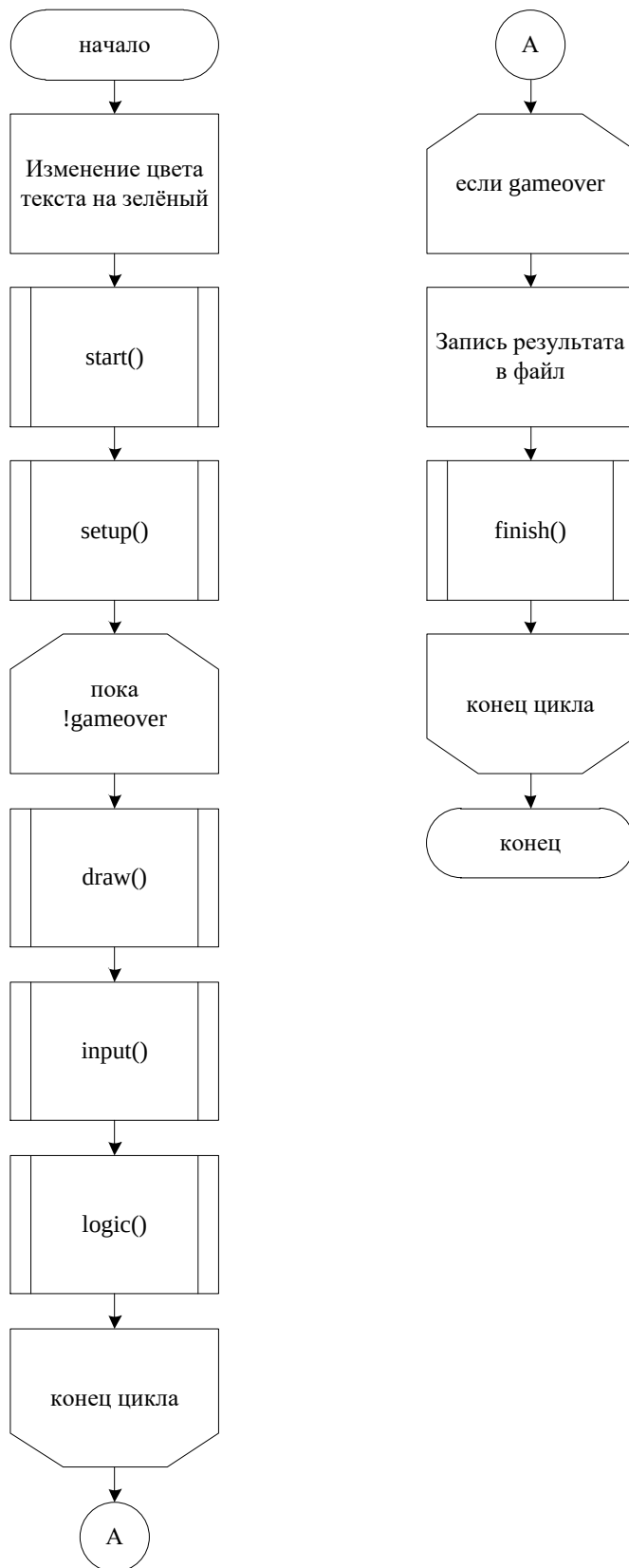


Рисунок 1 — Функция `main`

Приложение Snake.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. Вводится имя игрока с клавиатуры. Задаётся значение для координатных переменных. Задаются координаты для фрукта таким образом, чтобы они каждый раз принимали случайное значение. Добавляются переменные для подсчета очков и количества фруктов на карте. Создаётся массив для отображения поля игры.

createFruit – создание нового фрукта.

Draw - отрисовка карты, змейки и фрукта. Сначала очищается экран консоли, затем в массив записываются пустое поле, координаты головы и хвоста змейки, координаты фрукта. После этого выводим получившийся массив построчно. Под ним отображаем счётчик.

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

Logic - отвечает за логику всей игры. Осуществляем перемещение змейки (ячейки головы и хвоста). Разрабатываемая функция заканчивает игру, когда объект сталкивается с границей поля или сам с собой. Когда координаты головы объекта совпадают с координатами фрукта, длина хвоста объекта увеличивается на один символ, а к значению счётчика добавляется 10 очков. Каждые 50 очков увеличивается скорость змейки и количество яблок, также изменяется цвет символов консоли.

В файле «stfn.cpp» находятся функции start() и finish() , которые отображают заставку и окончание игры в консоли.

5.2 Разработка функций на языке Assembler

Как известно, функции, реализованные на языке ассемблера, занимают меньше места и работают быстрее. В качестве разрабатываемой функции была выбрана функция, заканчивающая игру, если «змейка» врывается в границу поля, а также функция, вычисляющая координаты первоначального расположения «змейки».

Описание функций выполнено на языке С, а вычисления (частично, в некоторых случаях приходилось использовать язык СИ) представляют собой ассемблерную вставку.

```
74  |  __asm {  
75  |      mov eax, height  
76  |      mov ebx, width  
77  |      sar eax, 1  
78  |      sar ebx, 1  
79  |      mov x, eax  
80  |      mov y, ebx  
81  |  }
```

Рисунок 2 — Ассемблерная вставка

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 `dec` уменьшает число на единицу;
- команда `cmp` используется для сравнения двух операндов;
- команда `jbe` выполняет короткий переход, если первый операнд МЕНЬШЕ или РАВЕН второму операнду при выполнении операции сравнения с помощью команды `cmp`;
- команда `jae` выполняет короткий переход, если первый операнд БОЛЬШЕ или РАВЕН второму операнду при выполнении операции сравнения с помощью команды `cmp`;
- команда `jmp` – команда безусловного перехода в Ассемблере;
- команда `xor` выполняет операцию исключающего ИЛИ между всеми битами двух операндов;
- команда `sar` выполняет арифметический сдвиг вправо

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

```

162  __asm {
163      mov eax, x
164      mov ebx, y
165      mov ecx, height
166      mov edx, width
167
168      dec ecx          //height - 1
169      dec edx          //width - 1
170      cmp ebx, 0       // если y <= 0
171      jbe l1
172      cmp ebx, ecx     // если y >= (height - 1)
173      jae l1
174      cmp eax, 0       // если x <= 0
175      jbe l1
176      cmp eax, edx     // если x >= (width - 1)
177      jae l1
178      jmp l2
179  l1 :
180      xor eax, eax     // eax = 0
181      mov eax, 1
182      mov gameover, eax // gameover = 1
183  l2 :

```

Рисунок 3 — Ассемблерная вставка

Переменные height и width уменьшаем на единицу. Если переменная y меньше или равна нулю или переменная y больше или равна height – 1 или переменная x меньше или равна нулю или переменная x больше или равна width – 1: игра заканчивается.

6. Руководство пользователя

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

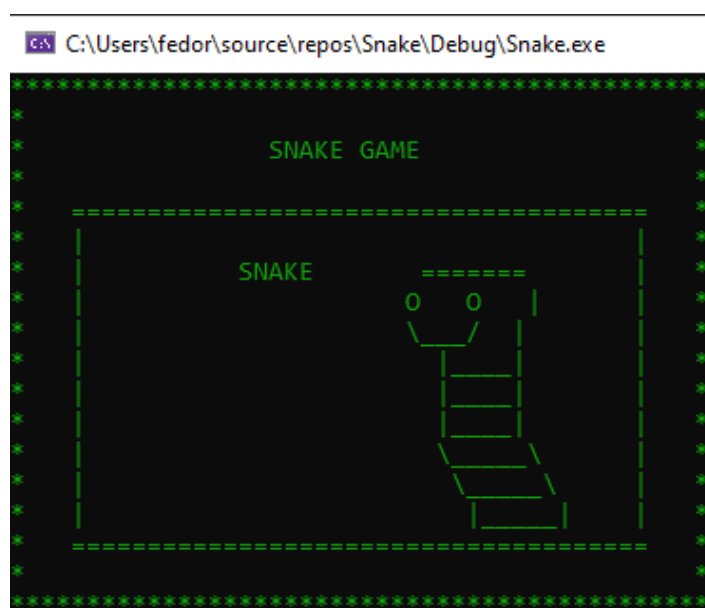


Рисунок 4.1 — Заставка

Вводится имя игрока и нажимается Enter:

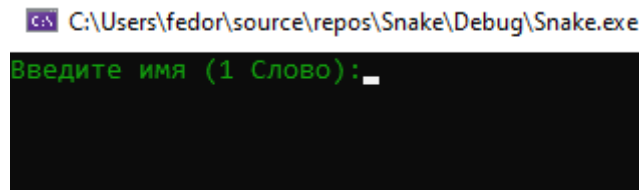


Рисунок 4.2 — Ввод имени пользователя

Отображается поле игры, голова змейки, случайным образом расположен фрукт, внизу находится счётчик, а также содержится информация о клавише, при нажатии которой игра завершится:

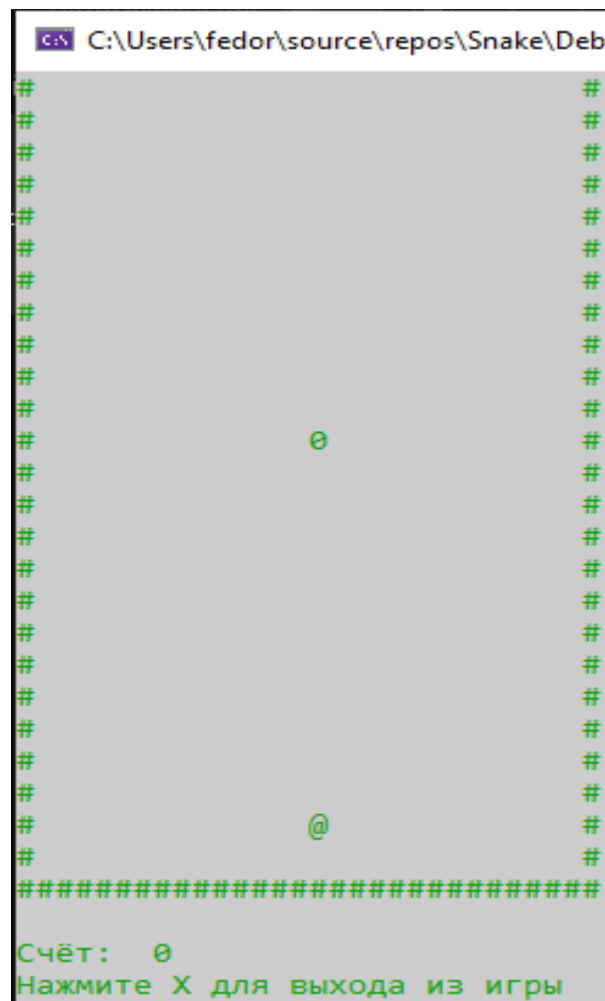


Рисунок 4.3 — Игровое поле

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

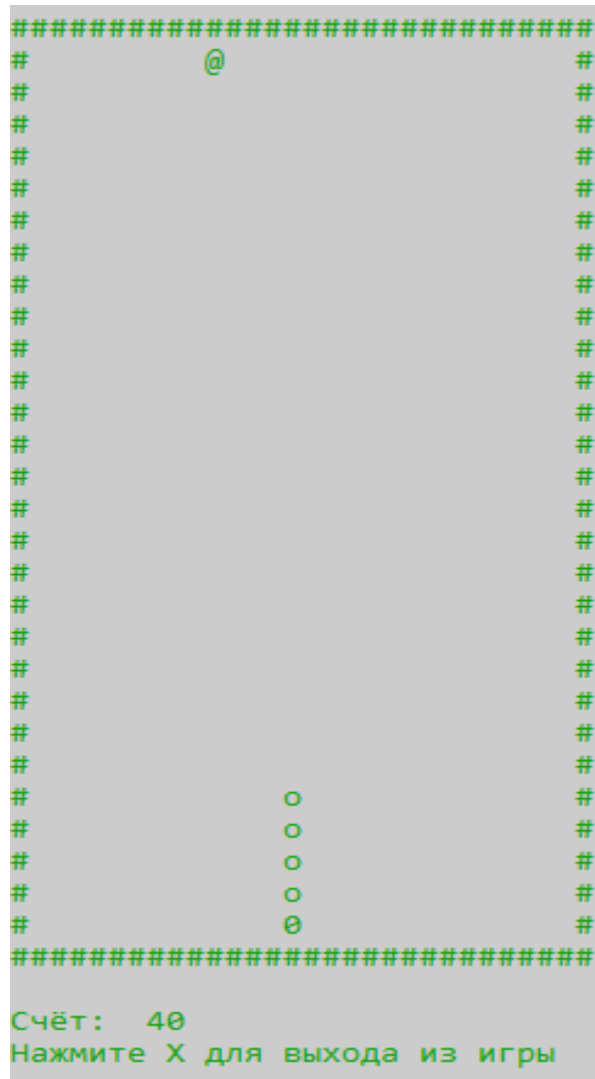


Рисунок 4.4 — Ход игры

Каждые 50 единиц изменяется цвет консоли, скорость змейки и количество плодов (максимальное количество плодов - 10):

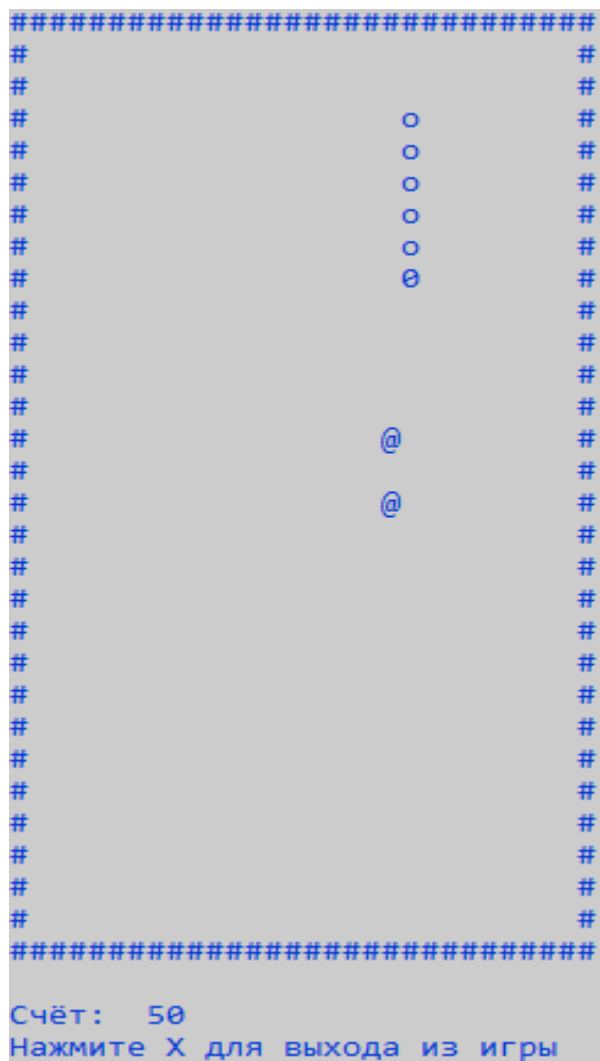


Рисунок 4.5 — Конец игры

После проигрыша появляется надпись «GAME OVER»



Рисунок 4.6 — Конец игры

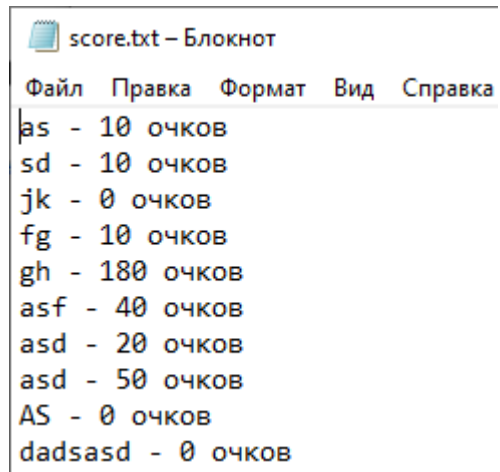


Рисунок 4.7 — Просмотр истории игр

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 2022. 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. Аблязов, Р. З. Программирование на ассемблере на платформе x86-64. / Р. З. Аблязов. – Москва : ДМК Пресс, 2011. – 304 с.
2. Жуков А. Ассемблер. Самоучитель / А. Жуков. – Санкт-Петербург : БХВ-Петербург, 2017. – 321 с. – ISBN 978-5-94157-133-8
3. Вострухин, А.В. Введение в программирование микроконтроллера AVR на языке Ассемблера / А.В. Вострухин, Е.А. Вахтина. – Москва : Илекса, 2010. – 184 с.
4. Васильев А. Н. Программирование на C++ в примерах и задачах / А. Н. Васильев. – Москва : Издательство "Э", 2017. – 368 с.
5. Павловская Т.А, C/C++ Программирование на языке высокого уровня / Т.А, Павловская. – ил. – СПб : Питер, 2007. – 461 с. – ISBN 978-5-94723-568-5

Приложение А. Иерархическая структура программы

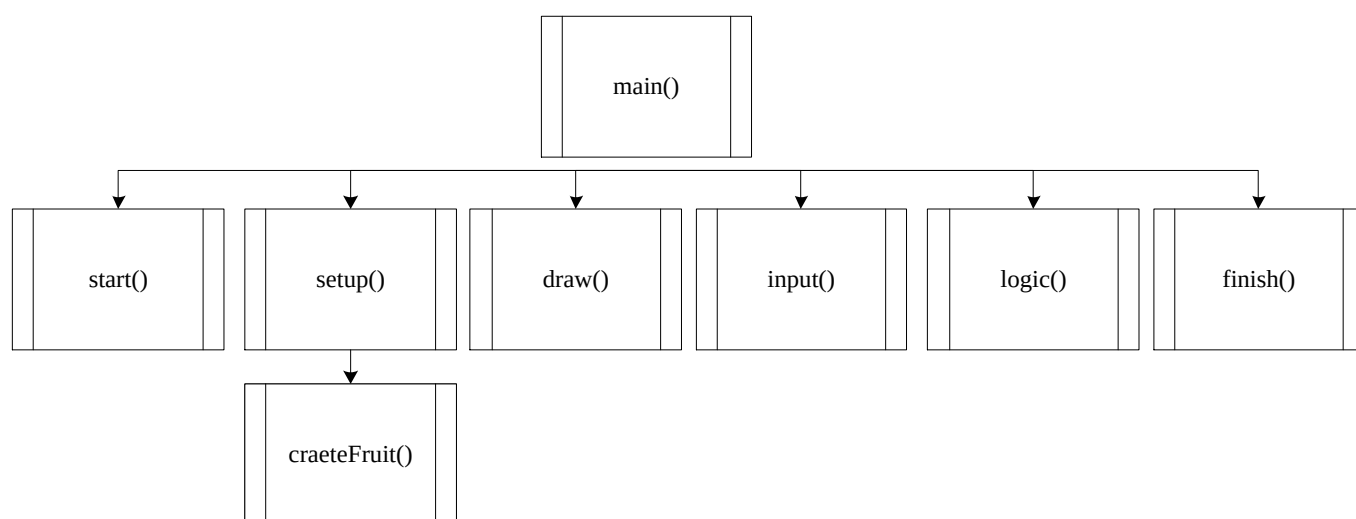


Рисунок 5 — Иерархическая структура программы

Приложение В. Схема данных программы

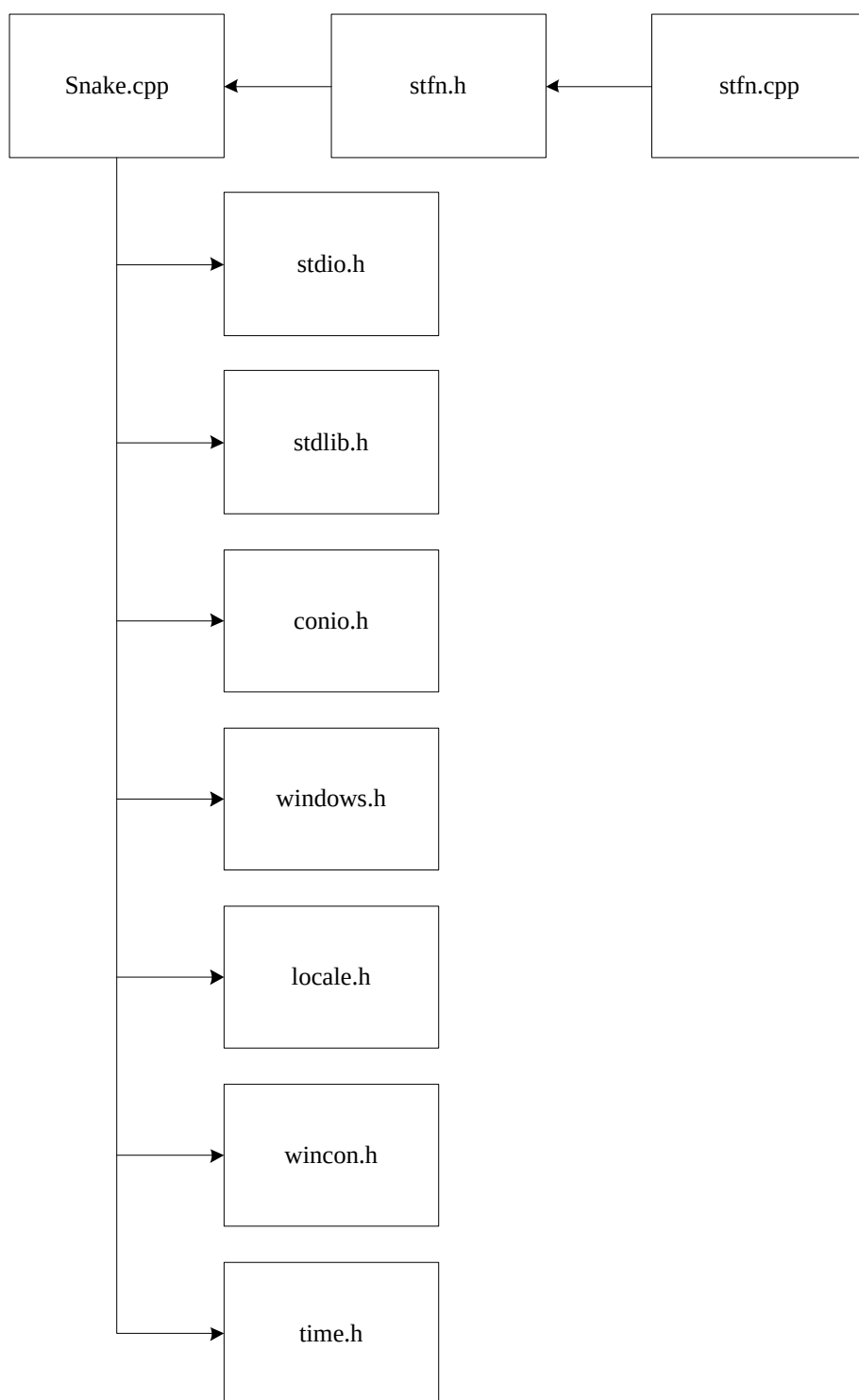


Рисунок 6 — Схема данных программы

Приложение С. Листинг программы

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <windows.h>
#include <locale.h>
#include <wincon.h>
#include <time.h>
#include "stfn.h"
//каждые 50 яблок меняется количество плодов, скорость, цвет

int i, j, gameover, x, y, fruitlx, fruitly, score, flag, t = 50,
sl = 500;
int const height = 30, width = 30;
int tailx[100], taily[100], tail;
int fruits[10][2], fruitcount;
char pixels[height][(width + 1)];
char nickname[100];

//создание поля
void draw() {
    system("cls"); //- удаляется всё с консоли

    printf(" \n");

    for (int i = 1; i < height - 1; i++) {
        for (int j = 1; j < width - 1; j++) {
            pixels[i][j] = ' ';
        }
    }
    pixels[0][0] = '#';

    pixels[y][x] = '0';
    for (int k = 0; k < tail; k++) {
        pixels[taily[k]][tailx[k]] = 'o';
    }
    for (int z = 0; z < fruitcount; z++) {
        pixels[fruits[z][1]][fruits[z][0]] = '@';
    }

    for (int i = 0; i < height; i++) {
        printf("%s\n", pixels[i]);
    }

    printf("\n");
    printf("Счёт: % d\n", score);

    printf("Нажмите X для выхода из игры\n");
    //printf("x:%d y:%d", x, y);
```

```

}

void createFruit() {
    for (int i = 0; i < fruitcount; i++) {
        fruits[i][0] = (rand() % (height - 2)) + 1;
        fruits[i][1] = (rand() % (height - 2)) + 1;
    }
}

//задаём положение змейки и "яблок"
void setup() {

    printf("Введите имя (1 Слово):");
    scanf("%s", nickname);
    system("cls");

    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
    FOREGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_RED |
    BACKGROUND_GREEN);

    srand(time(NULL));
    rand();
    fruitcount = 1;
    gameover = 0;

    __asm {
        mov eax, height
        mov ebx, width
        sar eax, 1
        sar ebx, 1
        mov x, eax
        mov y, ebx
    }
    /*x = height / 2;          y = width / 2;
    createFruit();
    score = 0;

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

    for (int i = 0; i < width; i++) { // Заполнение верхней и нижней
части поля
        pixels[0][i] = '#';
        pixels[(height - 1)][i] = '#';
    }
    for (int i = 0; i < height; i++) { // Заполнение правой и левой
стенки
        pixels[i][0] = '#';
        pixels[i][width - 1] = '#';
        pixels[i][width] = '\\0';
    }
}

```

```

}

//прием ввода с клавиатуры
void input() {
    dif (_kbhit()) { //для нажатия с клавиатуры
        switch (_getch()) { //switch - упрощение для if
            case 'a': //getch - чтение символа с клавиатуры.
                Символ не выводится
                flag = 1;
                break;
            case 's':
                flag = 2;
                break;
            case 'd':
                flag = 3;
                break;
            case 'w':
                flag = 4;
                break;
            case 'x':
                gameover = 1;
                break;
        }
    }
}

//основная логика программы
void logic() {
    int preX = tailx[0];
    int preY = taily[0];
    int pre2X, pre2Y;
    tailx[0] = x;
    taily[0] = y;
    for (int i = 1; i < tail; i++) {
        pre2X = tailx[i];
        pre2Y = taily[i];
        tailx[i] = preX;
        taily[i] = preY;
        preX = pre2X;
        preY = pre2Y;
    }

    Sleep(sl); //задержать обновление поля
    switch (flag) { //связываем клавиши с движением змейки
        case 1:
            x--;
            break;
        case 2:
            y++;
            break;
        case 3:
            x++;
            break;
        case 4:

```

```

        y--;
        break;
default:
        break;
}

// если игра заканчивается

__asm {
    mov eax, x
    mov ebx, y
    mov ecx, height
    mov edx, width

    dec ecx                //height - 1
    dec edx                //width - 1
    cmp ebx, 0             // если y <= 0
    jbe l1
    cmp ebx, ecx           // если y >= (height - 1)
    jae l1
    cmp eax, 0             // если x <= 0
    jbe l1
    cmp eax, edx           // если x >= (width - 1)
    jae l1
    jmp l2
l1:
    xor eax, eax           // eax = 0
    mov eax, 1
    mov gameover, eax      // gameover = 1
l2:
}
/*if (y <= 0 || y >= (height - 1) || x <= 0 || x >= (width - 1))
{
    gameover = 1;
}*/

for (int i = 0; i < tail; i++) {
    if (tailx[i] == x && taily[i] == y) {
        gameover = 1;
    }
}

for (int i = 0; i < fruitcount; i++) {
    if (x == fruits[i][0] && y == fruits[i][1]) { //если змея
съест плод
        score += 10; // увеличение счёта
        tail++;

        if (score == t) {
            fruitcount++;
            if (sl > 50) {
                sl -= 50;
            }
            if ((t / 50) % 2 == 0) {

```

```

        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_RED |
BACKGROUND_GREEN);
    }
    else {

        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_BLUE | BACKGROUND_BLUE | BACKGROUND_RED |
BACKGROUND_GREEN);
    }
    t += 50;
}
createFruit();
}
}

int main() {
    setlocale(LC_ALL, "Russian");
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_GREEN);

    start(); //заставка

    setup(); //создаём поле

    while (!gameover) { //пока игра не закончится

        draw();
        input();
        logic();
    }
    if (gameover) {

        const char* path = "score.txt";

        FILE* fp;
        fp = fopen(path, "a+");
        fprintf(fp, "%s - %d очков\n", nickname, score);
        fclose(fp);

        Sleep(3000);
        system("cls");
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_RED | BACKGROUND_BLUE | BACKGROUND_RED |
BACKGROUND_GREEN);

        finish();

        _getch();
        return 0;
    }
}

```


Приложение D. Схемы функций программы

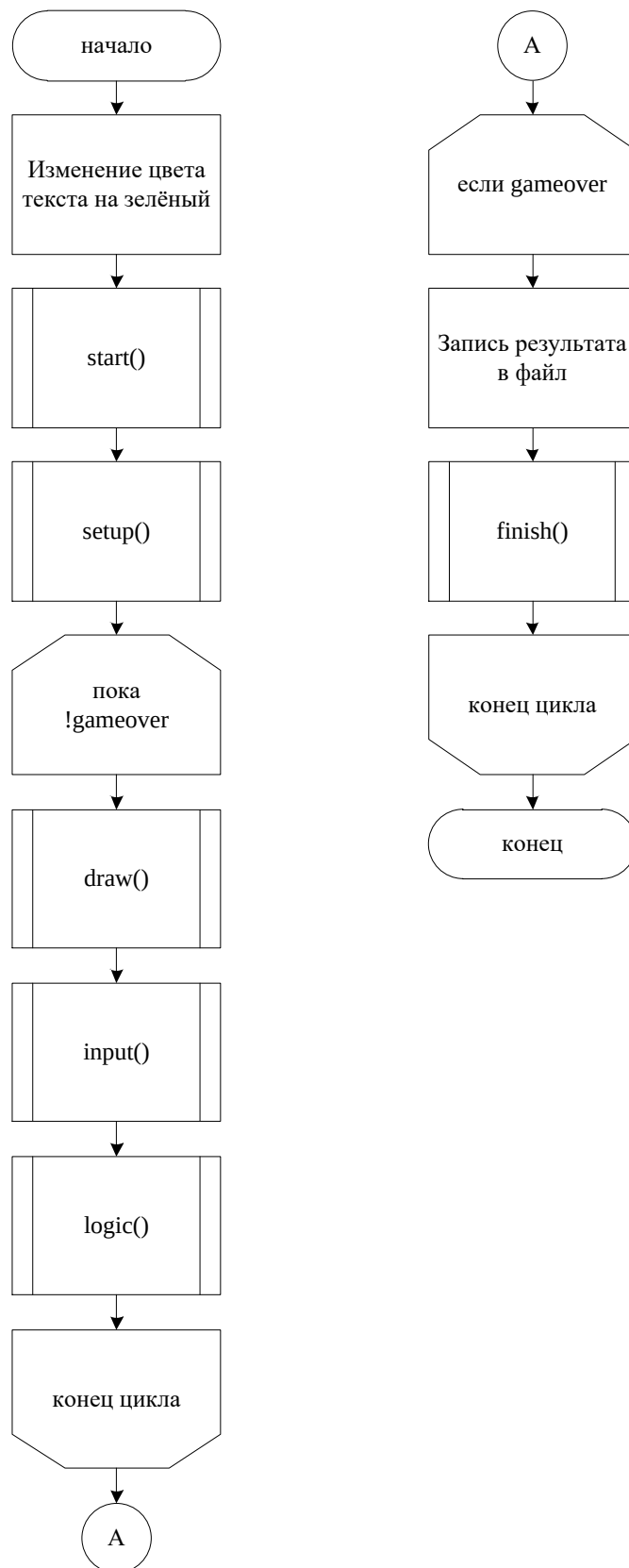


Рисунок 8.1 — Функция `main`

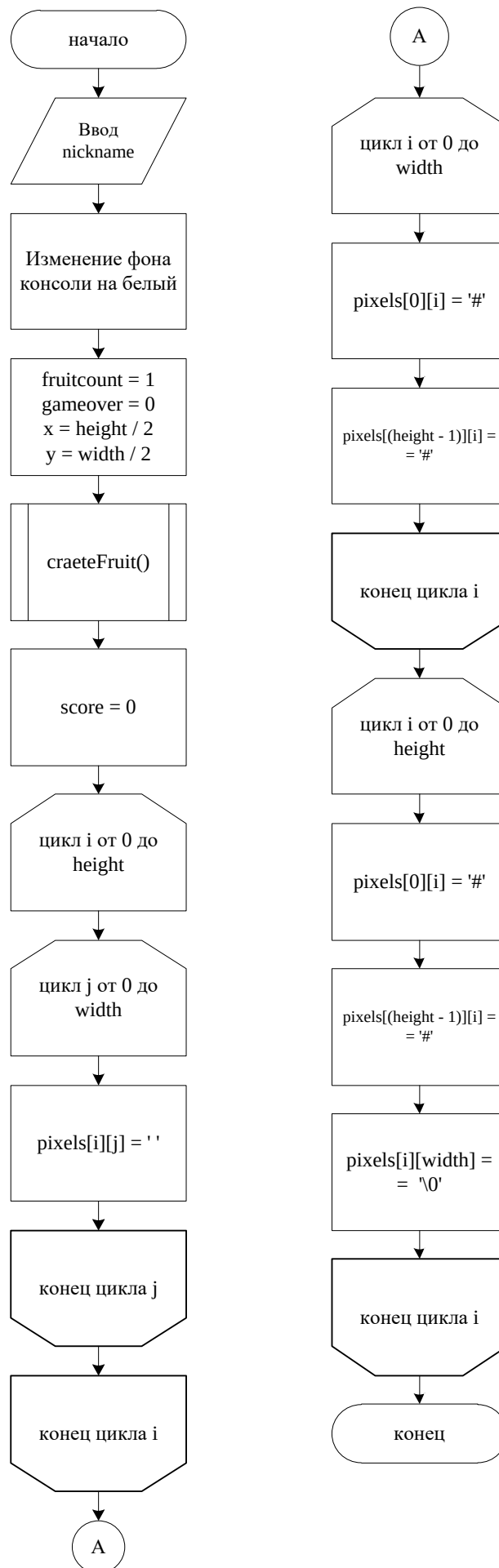


Рисунок 8.2 — Функция setup

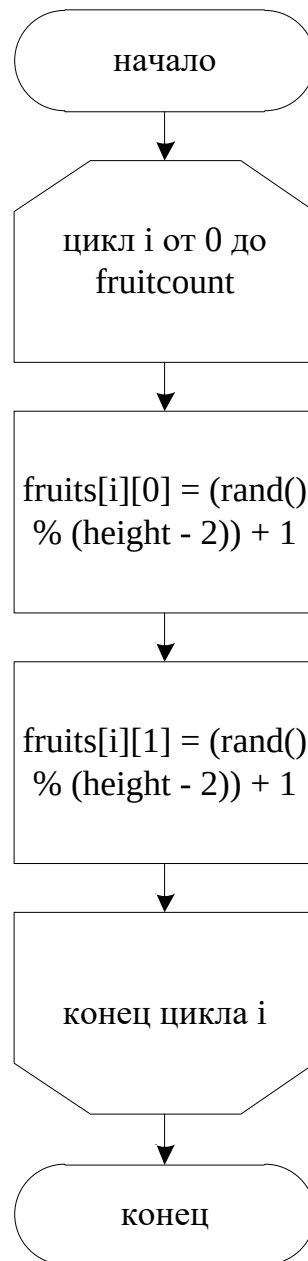


Рисунок 8.3 — Функция createFruit

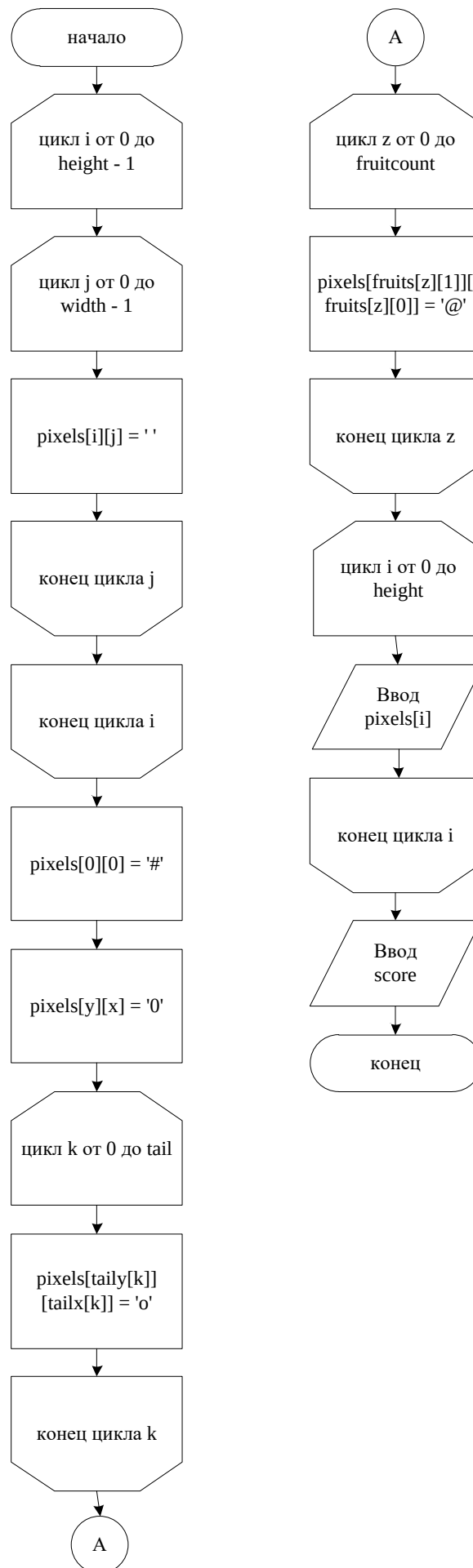


Рисунок 8.4 — Функция Draw

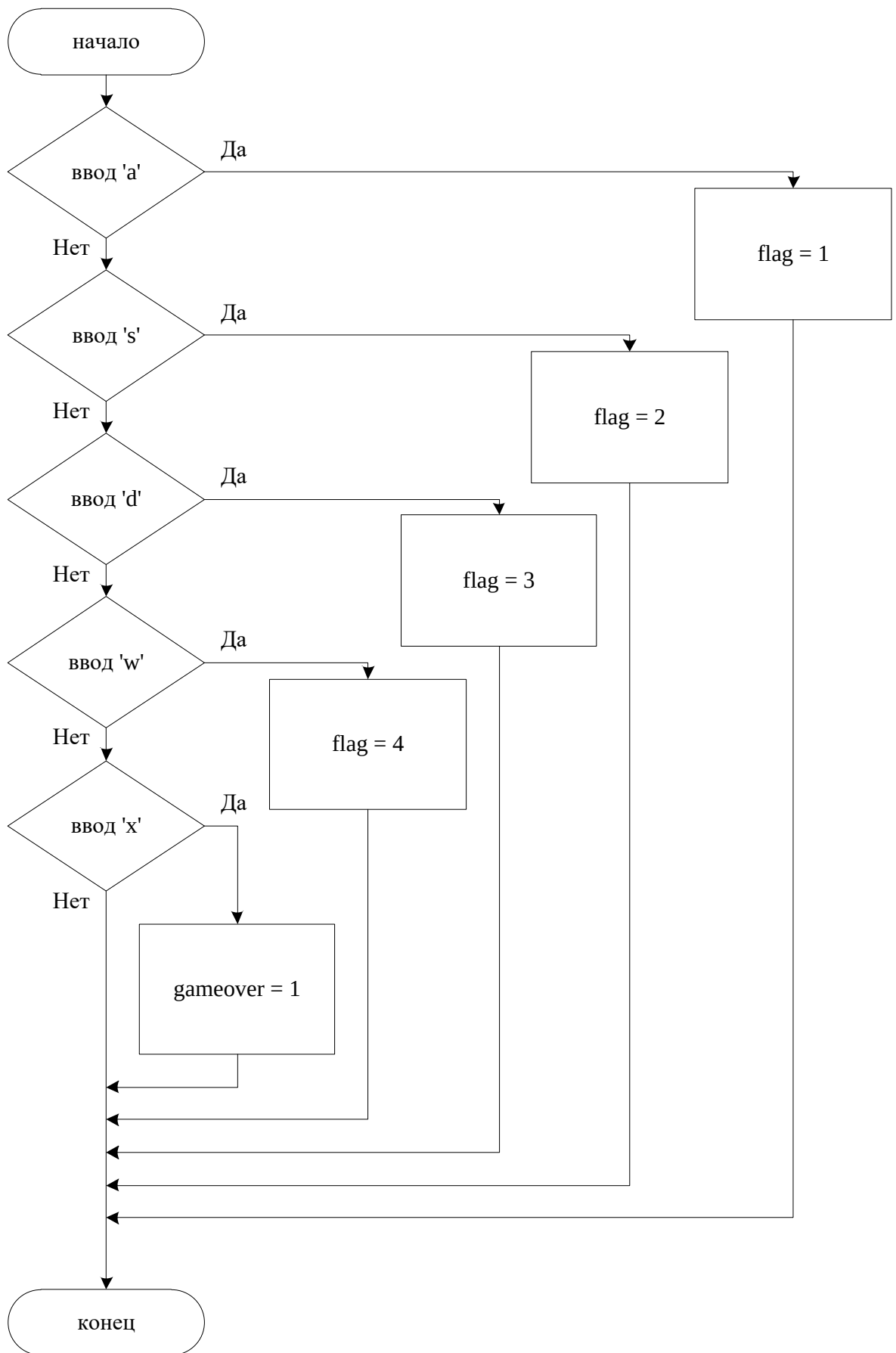


Рисунок 8.5 — Функция Input

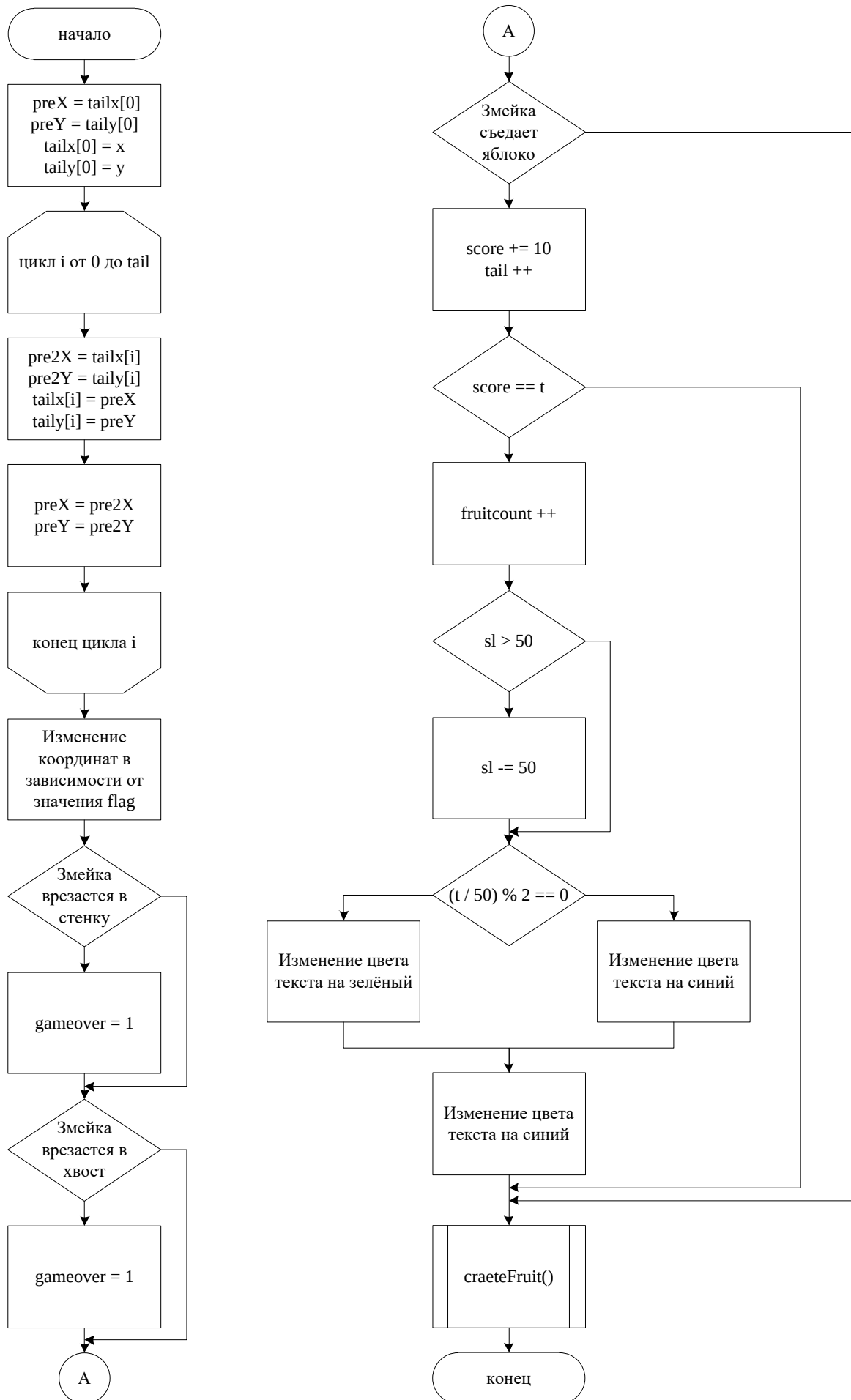


Рисунок 8.6 — Функция Logic

Приложение Е. Схема взаимодействия программы

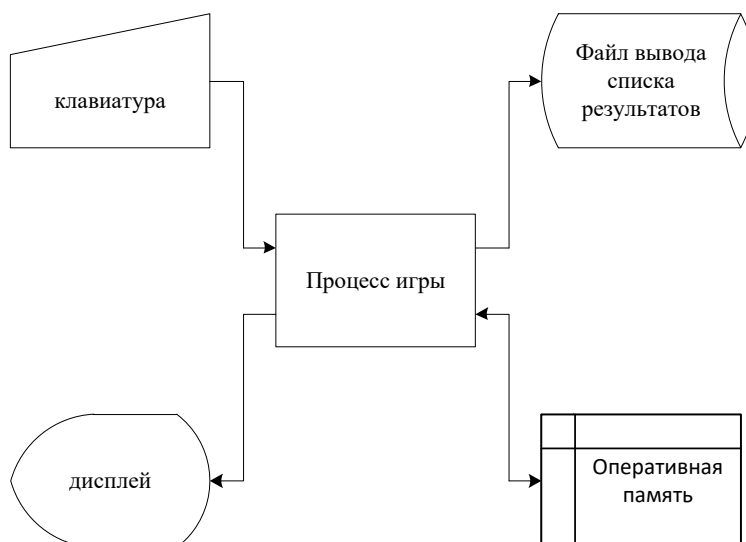


Рисунок 9 — Схема взаимодействия программы