

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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

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

Выполнил:

студент группы 23ВВВЗ

Андреева-Дубова А.В

удовл.
09.06.24

Принял:

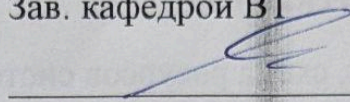
к.т.н., доцент Голотенков Н. О.

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 Андреевой-Дубовой Анастасии Группа 23ВВВ3

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

Исходные данные (технические требования) на проектирование

Разработать игру Tetris управление осуществляется с клавиатуры, игровое поле минимум 10 столбцов и 20 строк, по окончании игры вывести счет 10-ти лучших игроков.

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 "06" февраля 2024

Дата защиты проекта "04" 06.02

Руководитель Голотенков Н.О.

Задание получил "06" февраля 2024г.

Студент Андреева-Дубова А.В.

Содержание

Введение	5
Постановка задачи	6
Выбор решения	7
Описание разработки программы	7
Отладка и тестирование	8
Описание программы	8
Руководство пользователя	25
Заключение	29
Список используемых источников	30
Приложение А	31
Листинг программы	31
Приложение В	50

Введение

Игра тетрис – это одна из самых популярных и знаменитых игр в мире. Она была создана в 1984 году российским программистом Алексеем Пажитновым и стала настоящим хитом на протяжении многих лет. Сегодня тетрис доступен на всех платформах – от компьютеров до мобильных устройств. Игрокам предстоит собирать разноцветные блоки, которые падают сверху, и строить из них линии. Чем больше линий удалось собрать, тем больше очков получает игрок. Но игра не так проста, как может показаться на первый взгляд – блоки падают все быстрее и быстрее, и игрок должен быть очень быстр и внимателен, чтобы не допустить ошибки. Тетрис – это игра, которая требует от игрока не только ловкости и реакции, но и стратегического мышления. Она подходит для всех возрастов и уровней подготовки, и каждый найдет в ней что-то для себя.

Тетрис также имеет множество вариаций и модификаций, которые добавляют новые элементы и правила в игру. Некоторые из них включают в себя движение блоков в разных направлениях, возможность поворота блоков на

360 градусов или использование специальных блоков с уникальными свойствами. Кроме того, тетрис часто используется в качестве инструмента для тренировки мозга и улучшения когнитивных способностей. Многие исследования показали, что игра в тетрис может помочь улучшить концентрацию, реакцию и способность быстро принимать решения. В целом, тетрис – это игра, которая не только забавна и увлекательна, но и полезна для развития умственных способностей.

Постановка задачи

Необходимо разработать программу для игры в тетрис.

Программа должна иметь интуитивно понятный интерфейс для пользователя. Необходимо определить названия пунктов меню, построить алгоритмы работы с массивами структур для фильтрации и сортировки записей по определенным критериям, изучить функции для работы с файлами, чтобы сохранять базу данных в файл и загружать из файла.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы.

Использование сложных типов данных – массивов, структур, файлов. Это необходимо для более простой и интуитивной обработки данных в коде программы.

Режим работы видеосистемы – текстовый/графический. Для начала необходимо определиться с типом интерфейса и с элементами управления, затем необходимо изучить способы их реализации.

Устройство ввода-вывода – клавиатура и мышь. Необходимо различать и идентифицировать действия, произведенные с их помощью, это облегчит использование программы. Интерфейс должен быть построен на основе меню. Это необходимо для создания интуитивно понятного интерфейса для пользователя.

Заставка необходима для того, чтобы пользователь, запустивший программу, смог получить достаточную информацию о ней.

Выбор решения

При запуске программы открывается приветственный экран. Далее появляется меню, состоящее из трёх пунктов:

- 1) Начать игру. Пользователю выводится игровое поле, где он сможет увидеть.
- 2) Таблица рекордов. Пользователю выводится таблица с 10 лучшими результатами, которые записываются и считываются из файла rec.dat.
- 3) Выход. Выход из программы.

Навигация в меню осуществляется с помощью клавиатуры, нажатием на верхнюю и нижнюю стрелку.

Описание разработки программы

Разработанная программа состоит из нескольких модулей, которые реализовывались в следующем порядке:

- 1) Game.h – файл со всеми классами, необходимыми для работы программы.
- 2) Game.cpp – основной файл программы. Весь код с игрой тетрис находится внутри него.
- 3) Tetris.cpp – содержит заставку, создает переменную класса игры и запускает ее.

В качестве метода разработки программы было использовано ООП, где программа представлена в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определенного класса. Так в программе присутствует класс для всех фигур, класс для каждой отдельной фигуры, класс для работы с консолью, класс игры.

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft VisualStudio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей VisualStudio: точка останова, трассировка, анализ содержимого переменных. Тестирование проводилось во время разработки и также после завершения разработки. В ходе нее было выявлено огромное количество проблем, связанных с работой с файлами, работой с памятью, работой с полями структуры.

Описание программы

Приложение Tetris.exe является основным модулем программы. При запуске программы происходит вывод заставки-приветствия, после чего пользователю необходимо выбрать пункт меню для дальнейшего использования программы. Выход из меню осуществляется после выбора пользователем нужного пункта. Описание состояний программы указано ниже в таблицах:

Клавиши, вызывающее событие	Действие пользователя	Действие программы
Enter	Выбор 1 пункта меню “Начать игру”	Выполняется запуск игры(выполняет отображение поля игры)(управление см. Таблица 2.)
↓, Enter	Выбор 2 пункта меню “Таблица рекордов”	Программа выполняет отображение таблицы рекордов (Загружает ее из файла и выводит в консоль)

↓↓, Enter	Выбор 3 пункта меню “Выход”	Выполняется заккрытие программы
-----------	--------------------------------	------------------------------------

Таблица 1. – Взаимодействие с меню

Клавиши, вызывающее событие	Действие программы
←	Сдвиг фигуры влево
→	Сдвиг фигуры вправо
↑	Поворот фигуры
↓	Сдвиг фигуры вниз
Esc	Выход в главное меню

Таблица 2. – Управление в игре

Клавиши, вызывающее событие	Действие программы
Esc	Выход в главное меню

Таблица 3. – Клавиши взаимодействия в таблице рекордов

Руководство пользователя

Программа Tetris.exe предназначена для игры в тетрис. Программа поддерживает такие операции как управления клавишами, сохранение рекорда в файл, загрузку рекорда из файла.

В самом начале нас встречает заставка (рис. 1).

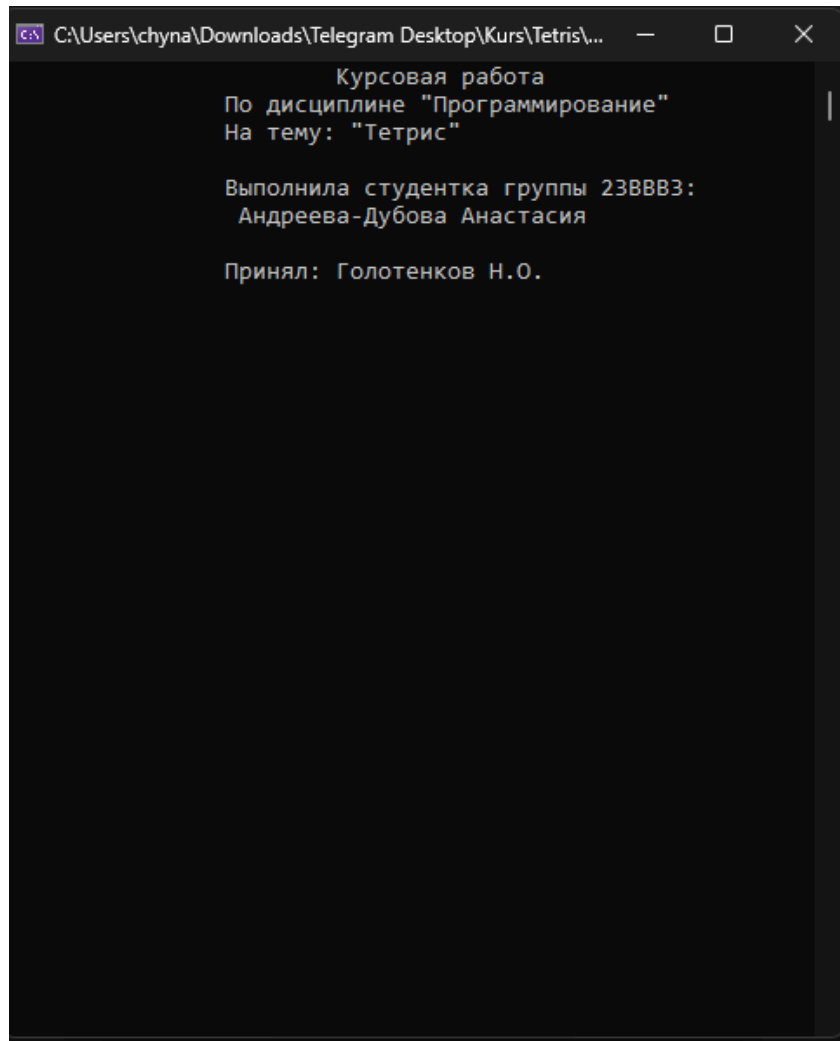


Рисунок 1 – Заставка приложения

После нажатия на Enter попадаем в меню игры (рис. 2).

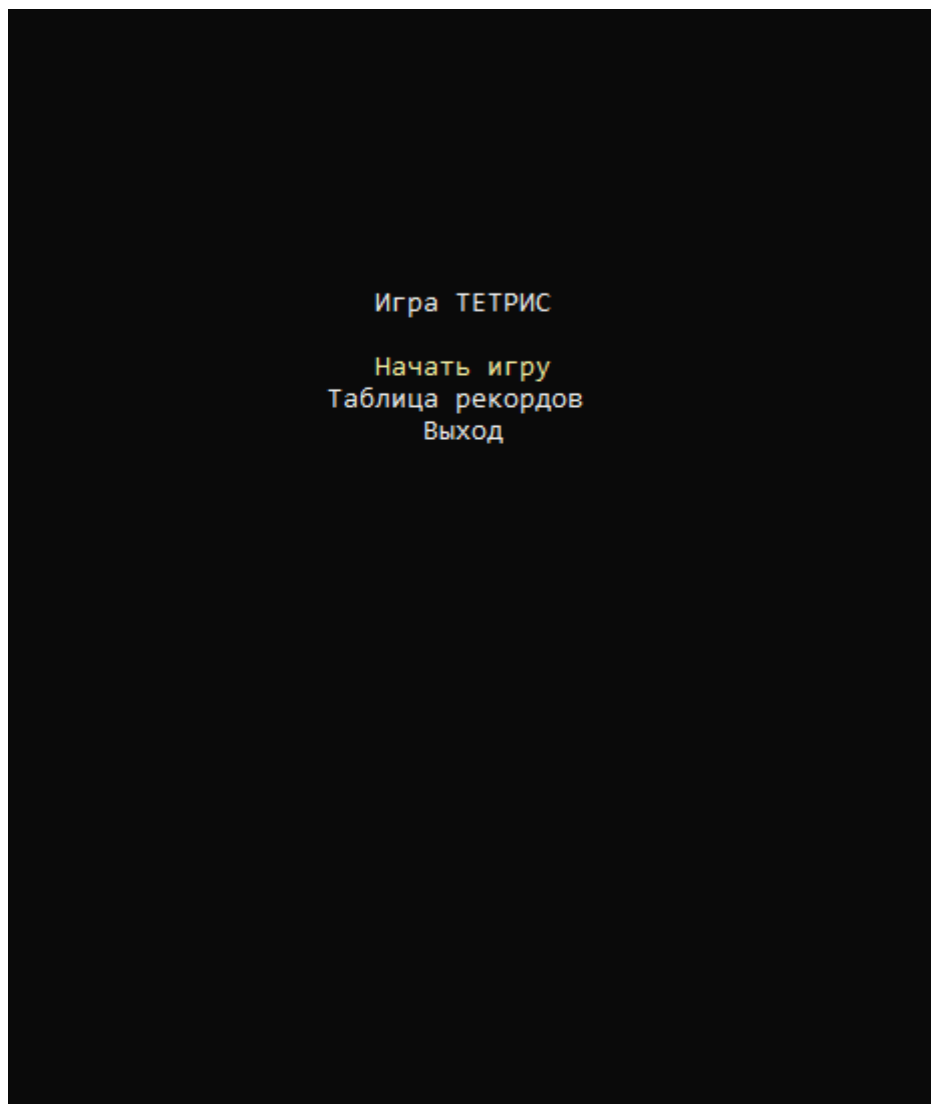


Рисунок 2 — Меню игры

Для начала игры нужно выбрать пункт «Начать игру» (Он выбран по умолчанию) и нажать Enter.

Далее отображается само поле игры в тетрис, очки и скорость игры (рис. 3)

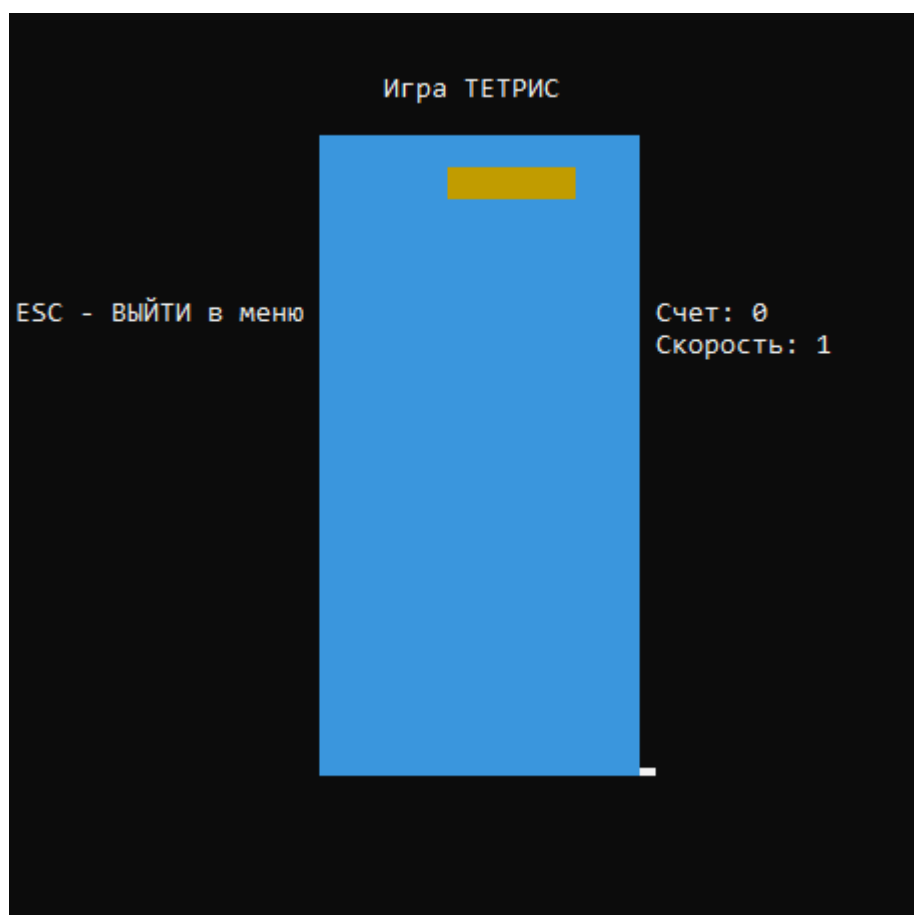


Рисунок 3— Игровое поле

Для управления фигурой в игре используются стрелки: вверх, повернуть, влево – сдвиг влево, вправо – сдвиг вправо, вниз - ускорить падение. Для увеличения счета на 100 очков необходимо стереть одну линию.

Скорость будет увеличиваться по мере игры тогда, когда разница нынешних очков и очков с момента последнего увеличения будет равна 3000.

Для того, чтобы вернуться в главное меню необходимо нажать Клавишу Esc, что указано на экране.

После возвращения в меню можно открыть таблицу рекордов. Для этого необходимо нажать «стрелку вниз» и «Enter». После чего откроется таблица с 10 лучшими рекордами.

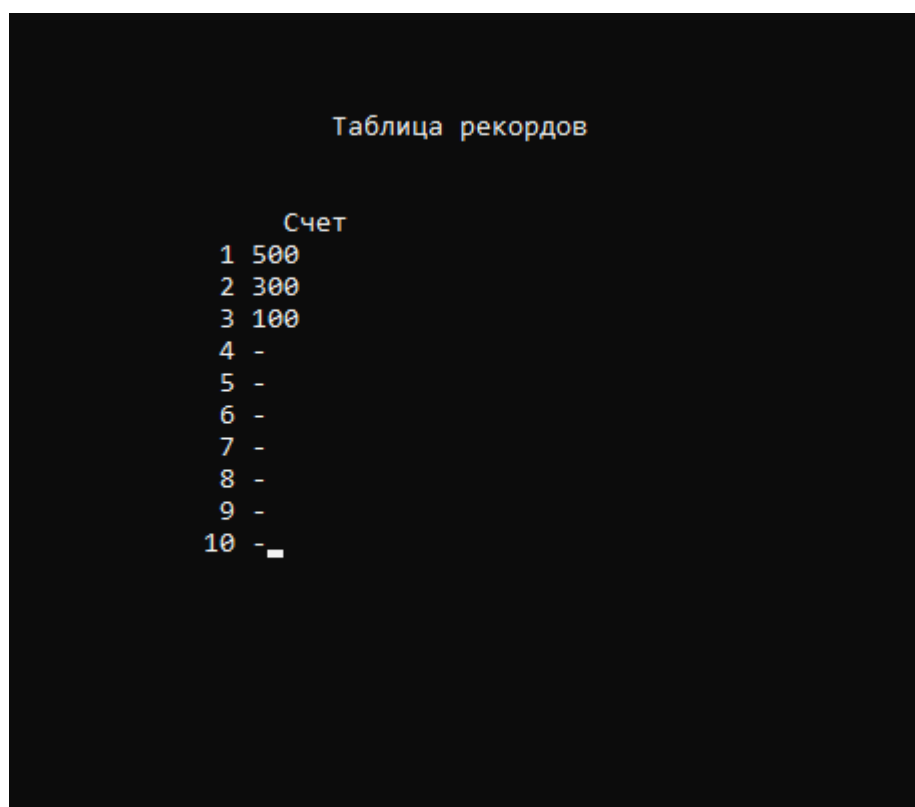


Таблица рекордов	
	Счет
1	500
2	300
3	100
4	-
5	-
6	-
7	-
8	-
9	-
10	-

Рисунок 4 — Таблица рекордов

Для выхода в главное меню снова необходимо нажать Esc. И после можно выбрать пункт «Выход» для закрытия программы.

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Была освоена работа с файлами, изучены функции работы с консолью и оперативной памятью. Также были получены основные навыки отладки и тестирования программ и программирования в среде VisualStudio 2022 на языках Си/C++ и Assembler. В рамках курсовой работы была написана программа для игры в тетрис. Она имеет все необходимые функции для работы с базами данных. В дальнейшем программу можно улучшить, путем добавления паузы во время самой игры, ввода имени при установлении рекорда, что улучшит взаимодействие пользователя с ней.

Список используемых источников

1. Герберт Шилдт. Полный справочник по С.
2. Н. Культин. Основы программирования в Microsoft Visual C++ 2010, 2010г
3. Харви и Пол Дейтел. Как программировать на С.
4. Александр Чиртик. Программирование на С++. Трюки и эффекты, 2010г.

Приложение А

Листинг программы

Файл Tetris.cpp

```
#include "Windows.h"
#include "Game.h"
#include <locale.h>
#include <time.h>

int main(int argc, wchar_t* argv[])
{
    setlocale(LC_ALL, "Russian"); // для нормального отображения русских букв
    srand((unsigned)time(NULL)); // инициализация случайных чисел текущим временем

    HWND hWindowConsole = GetConsoleWindow();
    RECT r;
    GetWindowRect(hWindowConsole, &r); // Запоминает текущие размеры консоли
    MoveWindow(hWindowConsole, r.left, r.top, 500, 600, TRUE); // Изменяет размер окна консоли

    wprintf(L"\t\t\tКурсовая работа\n\t\t\tПо дисциплине \t\t\t\"Программирование\"\n\t\t\tНа тему:
    \t\t\t\"Тетрис\"\n\n\t\t\tВыполнила студентка группы 23BBB3:\n\t\t\tАндреева-Дубова Анастасия\n\n\t\t\tПринял:
    Голотенков Н.О.");
    getchar();
    system("cls");

    CGame game; // Переменная класса игры
    game.run(); // Выполнение главного меню

    return 0;
}
```

Файл Game.h

```
#pragma once
```

```
#include <wtypes.h>
```

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

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

```
// Класс для работы с консолью
```

```
class CDisp {
```

```
protected:
```

```
HANDLE hnd; // Дескриптор консоли
```

```
public:
```

```
CDisp(); // Конструктор
```

```
void clear(); // Очистка экрана
```

```
void col(int text, int back); // Установка цвета текста и фона
```

```
void pos(int x, int y); // Установка позиции курсора
```

```
void win();
```

```
};
```

```
// Базовый класс для всех фигур
```

```
class CFigura {
```

```
protected:
```

```
int fi, fj; // Строка и столбец
```

```
int fr; // Вращение фигуры
```

```
public:
```

```
CFigura(); // Конструктор
```

```
void setpos(int i, int j); // Установка положения фигуры
```

```
void sdvig(int si, int sj); // Сдвиг фигуры
```

```
void read(FILE* f);
```

```
virtual void item(int i, int& pi, int& pj) = 0; // Получение координат i-й клетки фигуры
```

```
virtual int color() = 0; // Получение цвета фигуры
```

```
virtual void povorot(bool vpered) = 0; // Поворот фигуры (вперед или назад)
virtual int tip() = 0; // Тип фигуры: 0..6
};
```

// Фигура - квадрат

```
class CKvadFig : public CFigura {
public:
    CKvadFig(); // Конструктор
    virtual void item(int i, int& pi, int& pj); // Получение координат i-й клетки фигуры
    virtual int color() { return 2; } // Получение цвета фигуры
    virtual void povorot(bool vpered); // Поворот фигуры (вперед или назад)
    virtual int tip() { return 0; } // Тип фигуры
};
```

// Фигура - линия

```
class CLinFig : public CFigura {
public:
    CLinFig(); // Конструктор
    virtual void item(int i, int& pi, int& pj); // Получение координат i-й клетки фигуры
    virtual int color() { return 3; } // Получение цвета фигуры
    virtual void povorot(bool vpered); // Поворот фигуры (вперед или назад)
    virtual int tip() { return 1; } // Тип фигуры
};
```

// Фигура - зигзаг 1

```
class CZigz1Fig : public CFigura {
public:
    CZigz1Fig(); // Конструктор
    virtual void item(int i, int& pi, int& pj); // Получение координат i-й клетки фигуры
    virtual int color() { return 4; } // Получение цвета фигуры
    virtual void povorot(bool vpered); // Поворот фигуры (вперед или назад)
    virtual int tip() { return 2; } // Тип фигуры
};
```

// Фигура - зигзаг 2

```
class CZigz2Fig : public CFigura {
public:
    CZigz2Fig(); // Конструктор
    virtual void item(int i, int& pi, int& pj); // Получение координат i-й клетки фигуры
    virtual int color() { return 5; } // Получение цвета фигуры
};
```

```

virtual void povorot(bool vpered); // Поворот фигуры (вперед или назад)
virtual int tip() { return 3; } // Тип фигуры
};

// Фигура - кочерга 1
class CKoch1Fig : public CFigura {
public:
CKoch1Fig(); // Конструктор
virtual void item(int i, int& pi, int& pj); // Получение координат i-й клетки фигуры
virtual int color() { return 6; } // Получение цвета фигуры
virtual void povorot(bool vpered); // Поворот фигуры (вперед или назад)
virtual int tip() { return 4; } // Тип фигуры
};

// Фигура - кочерга 2
class CKoch2Fig : public CFigura {
public:
CKoch2Fig(); // Конструктор
virtual void item(int i, int& pi, int& pj); // Получение координат i-й клетки фигуры
virtual int color() { return 7; } // Получение цвета фигуры
virtual void povorot(bool vpered); // Поворот фигуры (вперед или назад)
virtual int tip() { return 5; } // Тип фигуры
};

// Фигура - пирамида
class CPirFig : public CFigura {
public:
CPirFig(); // Конструктор
virtual void item(int i, int& pi, int& pj); // Получение координат i-й клетки фигуры
virtual int color() { return 8; } // Получение цвета фигуры
virtual void povorot(bool vpered); // Поворот фигуры (вперед или назад)
virtual int tip() { return 6; } // Тип фигуры
};

// Запись таблицы рекордов
struct srec {
int ochki; // Очки
};

// Класс игры

```

```

class CGame {
public:
int matr[20][10]; // Игровое поле
CFigura* figur; // Текущая фигура
int ochki; // Очки
int ochki_pr; // Очки после последнего повышения скорости
int skor; // Скорость падения фигуры
int skor_v; // Время перемещения фигуры на шаг вниз
srec recs[10]; // Таблица рекордов
public:
CGame(); // Конструктор
~CGame(); // Деструктор
void newg(); // Начало новой игры
int menu(int k); // Запуск меню
void draw(); // Отображение игры
void game(bool _new); // Запуск игры (новой или продолжение)
void run(); // Запуск главного меню и выполнение действий по пунктам меню
bool testfigur(CFigura* figur); // Проверка возможности размещения фигуры
void addfigur(CFigura* figur); // Размещение фигуры в поле
void delfigur(CFigura* figur); // Удаление фигуры из поля
bool sdvig(int si, int sj); // Сдвиг фигуры
bool povorot(); // Поворот фигуры
bool newfigur(); // Случайный выбор следующей фигуры
void nachisl(); // Удаление линий и начисление очков
void readrec(); // Запись таблицы рекордов в файл
void writerec(); // Чтение таблицы рекордов из файла
void drawrec(); // Отображение таблицы рекордов
void pause();
};

```

Файл Game.cpp

```

#include "Game.h"
#include <sys/timeb.h>

CDisp disp; // переменная класса для работы с консолью

CDisp::CDisp() {

```

```

        hnd = GetStdHandle(STD_OUTPUT_HANDLE); // получение дескриптора
    }
    void CDisp::clear() { system("cls"); } // очистка экрана
    void CDisp::col(int text, int back) // установка цвета
    {
        SetConsoleTextAttribute(hnd, text + back * 16);
    }
    void CDisp::pos(int x, int y)
    {
        COORD c = { x,y };
        SetConsoleCursorPosition(hnd, c); // установка позиции курсора
    }
    void CDisp::win()
    {
    }
    CFigura::CFigura()
    {
        fr = 0;
    }
    // установка положения фигуры
    void CFigura::setpos(int i, int j)
    {
        fi = i; fj = j;
    }
    // сдвиг фигуры
    void CFigura::sdvig(int si, int sj)
    {
        fi += si; fj += sj;
    }
    // чтение из файла
    void CFigura::read(FILE* f)
    {
        fread(&fi, sizeof(fi), 1, f);
        fread(&fj, sizeof(fj), 1, f);
        fread(&fr, sizeof(fr), 1, f);
    }

```

```

CKvadFig::CKvadFig()
{
    fi = 0;
    fj = 5;
}
// получение координат i-й клетки фигуры
void CKvadFig::item(int i, int& pi, int& pj)
{
    pi = fi; pj = fj; switch (i) {
        case 1: pi++; break; case 2: pj++; break;
        case 3: pi++; pj++; break;
    }
}
// поворот фигуры (вперед или назад)
void CKvadFig::povorot(bool vpered) {}

CLinFig::CLinFig()
{
    fi = 0;
    fj = 5;
}
void CLinFig::item(int i, int& pi, int& pj)
{
    pi = fi; pj = fj; if (fr == 0) {
        switch (i) {
            case 0: pj--; break; case 2: pj++; break; case 3: pj += 2; break;
        }
    }
    else {
        switch (i) {
            case 0: pi--; break; case 2: pi++; break; case 3: pi += 2; break;
        }
    }
}
// поворот фигуры (вперед или назад)
void CLinFig::povorot(bool vpered)
{

```

```

        if (fr == 0) fr = 1; else fr = 0;
    }
    CZigz1Fig::CZigz1Fig()
    {
        fi = 0;
        fj = 5;
    }
    // получение координат i-й клетки фигуры
    void CZigz1Fig::item(int i, int& pi, int& pj)
    {
        pi = fi; pj = fj; if (fr == 0) {
            switch (i) {
                case 0: pj--; break; case 2: pi++; break;
                case 3: pi++; pj++; break;
            }
        }
        else {
            switch (i) {
                case 0: pi--; pj++; break; case 1: pj++; break;
                case 3: pi++; break;
            }
        }
    }

    // поворот фигуры (вперед или назад)
    void CZigz1Fig::povorot(bool vpered)
    {
        if (fr == 0)
            fr = 1;
        else
            fr = 0;
    }

    CZigz2Fig::CZigz2Fig()
    {
        fi = 0;

```

```

        fj = 5;
    }

    // получение координат i-й клетки фигуры
    void CZigz2Fig::item(int i, int& pi, int& pj)
    {
        pi = fi;
        pj = fj;
        if (fr == 0) {
            switch (i) {
            case 0:
                pi++;
                pj--;
                break;

            case 1:
                pi++;
                break;

            case 3:
                pj++;
                break;

            }
        }
        else {
            switch (i) {
            case 0:
                pi--;
                break;

            case 2:
                pj++;
                break;

            case 3:
                pi++;
                pj++;
                break;

            }
        }
    }

```

```
}
```

```
// поворот фигуры (вперед или назад)
```

```
void CZigz2Fig::povorot(bool vpered)
```

```
{
```

```
    if (fr == 0)
```

```
        fr = 1;
```

```
    else
```

```
        fr = 0;
```

```
}
```

```
CKoch1Fig::CKoch1Fig()
```

```
{
```

```
    fi = 0;
```

```
    fj = 5;
```

```
}
```

```
// получение координат i-й клетки фигуры
```

```
void CKoch1Fig::item(int i, int& pi, int& pj)
```

```
{
```

```
    pi = fi;
```

```
    pj = fj;
```

```
    if (fr == 0) {
```

```
        switch (i) {
```

```
        case 0:
```

```
            pj--;
```

```
            break;
```

```
        case 2:
```

```
            pj++;
```

```
            break;
```

```
        case 3:
```

```
            pi++;
```

```
            pj++;
```

```
            break;
```

```
        }
```

```
    }
```

```

else if (fr == 1) {
    switch (i) {
        case 0:
            pi--;
            break;

        case 2:
            pi++;
            break;

        case 3:
            pi++;
            pj--;
            break;

    }
}

else if (fr == 2) {
    switch (i) {
        case 0:
            pi--;
            pj--;
            break;

        case 1:
            pj--;
            break;

        case 3:
            pj++;
            break;

    }
}

else if (fr == 3) {
    switch (i) {
        case 0:
            pi--;
            break;

        case 1:
            pi--;
            pj++;

```

```

                                break;
                                case 3:
                                pi++;
                                break;
                                }
                                }
                                }

```

// поворот фигуры (вперед или назад)

void CKoch1Fig::povorot(bool vpered)

```

{
    if (vpered) { // вперед
        if (fr < 3)
            fr++;
        else
            fr = 0;
    }
}

```

CKoch2Fig::CKoch2Fig()

```

{
    fi = 0;
    fj = 5;
}

```

// получение координат i-й клетки фигуры

void CKoch2Fig::item(int i, int& pi, int& pj)

```

{
    pi = fi;
    pj = fj;
    if (fr == 0) {
        switch (i) {
            case 0:
                pi++;
                pj--;
                break;

```

```

        case 1:
            pj--;
            break;

        case 3:
            pj++;
            break;

    }
}

else if (fr == 1) {
    switch (i) {
        case 0:
            pi--;
            break;

        case 1:
            pi--;
            pj--;
            break;

        case 3:
            pi++;
            break;

    }
}

else if (fr == 2) {
    switch (i) {
        case 0:
            pj--;
            break;

        case 2:
            pj++;
            break;

        case 3:
            pi--;
            pj++;
            break;

    }
}

```

```

        else if (fr == 3) {
            switch (i) {
                case 0:
                    pi--;
                    break;

                case 2:
                    pi++;
                    break;

                case 3:
                    pi++;
                    pj++;
                    break;

            }
        }
    }
}

```

// поворот фигуры (вперед или назад)

```

void CKoch2Fig::povorot(bool vpered)
{
    if (vpered) { // вперед
        if (fr < 3)
            fr++;

        else
            fr = 0;
    }
}

```

CPirFig::CPirFig()

```

{
    fi = 0;
    fj = 5;
}

```

// получение координат i-й клетки фигуры

```

void CPirFig::item(int i, int& pi, int& pj)
{

```

```

pi = fi;
pj = fj;
if (fr == 0) {
    switch (i) {
        case 0:
            pj--;
            break;

        case 2:
            pj++;
            break;

        case 3:
            pi++;
            break;

    }
}
else if (fr == 1) {
    switch (i) {
        case 0:
            pi--;
            break;

        case 2:
            pi++;
            break;

        case 3:
            pj--;
            break;

    }
}
else if (fr == 2) {
    switch (i) {
        case 0:
            pi--;
            break;

        case 1:
            pj--;
            break;
    }
}

```

```

        case 3:
            pj++;
            break;
    }
}
else if (fr == 3) {
    switch (i) {
        case 0:
            pi--;
            break;

        case 1:
            pj++;
            break;

        case 3:
            pi++;
            break;
    }
}
}

```

// поворот фигуры (вперед или назад)

void CPirFig::povorot(bool vpered)

```

{
    if (vpered) { // вперед
        if (fr < 3)
            fr++;

        else
            fr = 0;
    }
    else { // назад
        if (fr > 0)
            fr--;

        else
            fr = 3;
    }
}

```

```

CGame::CGame()
{
    figur = NULL;
    ochki = 0;
}

CGame::~CGame()
{
    if (figur) {
        delete figur;
        figur = NULL;
    }
}

// проверка возможности размещения фигуры
bool CGame::testfigur(CFigura* figur)
{
    int pi, pj;
    for (int i = 0; i < 4; i++) {
        figur->item(i, pi, pj); // получение координат клетки
        if (pi >= 20 || pj < 0 || pj >= 10)
            return false;
        if (pi >= 0 && matr[pi][pj] != 0)
            return false;
    }
    return true;
}

// размещение фигуры в поле
void CGame::addfigur(CFigura* figur)
{
    int pi, pj;
    for (int i = 0; i < 4; i++) {
        figur->item(i, pi, pj); // получение координат клетки
        if (pi >= 0 && pi < 20 && pj >= 0 || pj < 10)

```

```

                                matr[pi][pj] = figur->color();
                                }
                                }

// удаление фигуры из поля
void CGame::delfigur(CFigura* figur)
{
    int pi, pj;
    for (int i = 0; i < 4; i++) {
        figur->item(i, pi, pj); // получение координат клетки
        if (pi >= 0 && pi < 20 && pj >= 0 || pj < 10)
            matr[pi][pj] = 0; // 0 - клетка свободна
    }
}

// начало новой игры
void CGame::newg()
{
    // очистка поля
    for (int i = 0; i < 20; i++)
        for (int j = 0; j < 10; j++)
            matr[i][j] = 0; // 0 - клетка свободна

    ochki = 0;
    ochki_pr = 0;
    skor = 1;
    skor_v = 1000;
    newfigur();
}

// запуск меню
int CGame::menu(int k)
{
    const char* line[] = {
        "  Начать игру  ",
        " Таблица рекордов ",

```

```

        " Выход"
};
disp.clear();
int n = k;
while (1) {
    disp.col(15, 0);
    disp.pos(23, 9);
    printf("Игра ТЕТРИС");
    for (int i = 0; i < 3; i++) // вывод строк меню
    {
        if (i == n)
            disp.col(14, 0); // цвет выделенной строки
        else
            disp.col(15, 0); // цвет невыделенной строки
        disp.pos(19, 11 + i); // позиция
        printf(line[i]); // вывод текста
    }
    disp.pos(30, 25);
    // ввод нажатой клавиши
    int c = 0;
    while (!_kbhit()) {}
    while (_kbhit())
        c = _getch();

    switch (c)
    {
    case 72: // курсор вверх
        if (n > 0)
            n--;

        break;
    case 80: // курсор вниз
        if (n < 3)
            n++;

        break;
    case 13:
        return n; // Enter
    }
}

```

```

    }
}

// отображение игры
void CGame::draw()
{
    // вывод счета
    disp.col(15, 0);
    disp.pos(41, 10);
    printf("Счет: %d", ochki);
    disp.pos(41, 11);
    printf("Скорость: %d", skor);
    // вывод игрового поля
    for (int i = 0; i < 20; i++)
    {
        disp.pos(20, 5 + i);
        for (int j = 0; j < 10; j++)
        {
            disp.col(50, matr[i][j]); // установка цвета фона
            printf(" "); // одна клетка поля - это 2 пробела
        }
    }
}

// сдвиг фигуры
bool CGame::sdvig(int si, int sj)
{
    bool res = true;
    delfigur(figur); // удаление фигуры из поля
    figur->sdvig(si, sj); // сдвиг
    if (!testfigur(figur)) { // если недопустимое положение фигуры
        res = false;
        figur->sdvig(-si, -sj); // сдвиг обратно
    }
    addfigur(figur); // возвращение фигуры на поле
    return res;
}

```

```

}

// поворот фигуры
bool CGame::povorot()
{
    bool res = true;
    delfigur(figur); // удаление фигуры из поля
    figur->povorot(true); // поворот вперед
    if (!testfigur(figur)) { // если недопустимое положение фигуры
        res = false;
        figur->povorot(false); // поворот обратно
    }
    addfigur(figur); // возвращение фигуры на поле
    return res;
}

// случайный выбор следующей фигуры
bool CGame::newfigur()
{
    if (figur)
        delete figur;
    int k = rand() % 7; // случайное число от 0 до 6
    // создание фигуры
    switch (k)
    {
    case 0:
        figur = new CKvadFig;
        break;
    case 1:
        figur = new CLinFig;
        break;
    case 2:
        figur = new CZigz1Fig;
        break;
    case 3:
        figur = new CZigz2Fig;

```

```

        break;
case 4:
        figur = new CKoch1Fig;
        break;
case 5:
        figur = new CKoch2Fig;
        break;
case 6:
        figur = new CPirFig;
        break;
}
bool res = testfigur(figur); // res - допустимость положения фигуры (не наложилась на другую)
addfigur(figur); // установить фигуру в поле draw();
if (res)
        return true; // если допустимо - вернуть true
// иначе игра закончена
disp.col(15, 0);
disp.pos(17, 13);
printf("Игра закончена! Ваши очки: %d", ochki);
getchar();
// занесение в таблицу рекордов
if (recs[9].ochki < ochki) {
        disp.pos(20, 14);
        for (int i = 0; i < 10; i++) {
                if (recs[i].ochki < ochki) { // если количество очков меньше, чем
набранное
                                                // сдвиг меньших результатов
                                                for (int j = 9; j > i; j--)
                                                        memcpy(&recs[j], &recs[j
- 1], sizeof(srec));
                                                // запись нового результата
                                                recs[i].ochki = ochki;
                                                writerec(); // сохранение таблицы результатов в
файл
                                                break;
                }
        }
}

```

```

    }
    else
        _getch();
    return false;
}

// удаление линий, если они заполнены и начисление очков
void CGame::nachisl() {
    int n = 0;
    for (int i = 19; i >= 0; i--) {
        bool f = true;
        for (int j = 0; j < 10; j++) {
            if (matr[i][j] == 0) { // в линии есть пустая клетка
                f = false; // неполная линия
                break;
            }
        }
        if (f)
            n++;
        else // удаленная линия (полная)
            if (n > 0) { // иначе, если были удаленные линии
                // копирование линии на n вниз и очистка
                for (int j = 0; j < 10; j++) {
                    matr[i + n][j] = matr[i][j];
                    matr[i][j] = 0;
                }
            }
    }
    switch (n) { // количество удаленных линий
        case 1: ochki += 100; break;
        case 2: ochki += 300; break;
        case 3: ochki += 700; break;
        case 4: ochki += 1500; break;
    }
    // если набрано 3000 очков после предыдущего увеличения скорости - повысить скорость
    (уменьшить время)
    int ochki1 = ochki;

```

```

int ochki_pr1 = ochki_pr;
int skor1 = skor;
int skor_v1 = skor_v;
int mn = 200;

_asm {
    mov eax, ochki1
    sub eax, ochki_pr1
    cmp eax, 3000
    jle end_label

    mov ecx, ochki1
    mov ochki_pr1, ecx
    inc skor1
    mov ebx, skor_v1
    sub ebx, 100
    mov skor_v1, ebx
}

end_label;

    ochki = ochki1;
    ochki_pr = ochki_pr1;
    skor = skor1;
    skor_v = skor_v1;
}

```

// запуск игры (новой или продолжение)

```

void CGame::game(bool _new) {
    disp.col(0, 0);
    disp.clear();
    if (_new) // новая игра
        newg(); // начать

    bool dr = true;
    int t1;

```

```

    _timeb tm;
    _ftime_s(&tm);
    t1 = (tm.time % 1000000) * 1000 + tm.millitm; // 6 цифр от количества секунд * 1000 +
количество миллисекунд
    disp.col(15, 0);
    disp.pos(24, 3);
    printf("Игра ТЕТРИС");
    disp.col(15, 0);
    disp.pos(1, 10);
    printf("ESC – ВЫЙТИ в меню");

    while (1) {
        _ftime_s(&tm);
        int t2 = (tm.time % 1000000) * 1000 + tm.millitm;
        if (t2 - t1 > skor_v) { // сдвиг вниз
            if (!sdvig(1, 0)) { // если сдвиг невозможен
                nachisl(); // начисление очков
                if (!newfigur()) { // появление новой фигуры,
если возможно
                return;
            }
        }
        dr = true;
        t1 = t2;
    }
    if (dr) { // нужно отобразить игру
        draw(); // отображение
        dr = false;
    }
    // ввод нажатой клавиши
    int c = 0;
    if (_kbhit())
        while (_kbhit())
            c = _getch();

    // обработка клавиш
    switch (c)

```

```

{
    case 75: if (sdvig(0, -1)) dr = true; // курсор влево - сдвиг фигуры влево
            break;
    case 77: if (sdvig(0, 1)) dr = true; // курсор вправо - сдвиг фигуры вправо
            break;
    case 72: if (povorot()) dr = true; // курсор вверх - поворот фигуры
            break;
    case 80: if (sdvig(1, 0)) dr = true; // курсор вниз - сдвиг вниз
            break;
    case 27: // Esc
            disp.col(15, 0);
            disp.pos(18, 13);
            printf("Успешный выход из игры!");
            _getch();
            return;
}
}
}

```

// загрузка таблицы рекордов

```

void CGame::readrec() {
    FILE* f;
    if (fopen_s(&f, "rec.dat", "rb")) { // открытие файла
        memset(recs, 0, sizeof(srec) * 10);
        return;
    }
    // чтение данных
    for (int i = 0; i < 10; i++)
        fread(&recs[i], sizeof(srec), 1, f);
    fclose(f);
}

```

// сохранение таблицы рекордов

```

void CGame::writerec() {
    FILE* f;

```

```

        if (fopen_s(&f, "rec.dat", "wb"))
            return; // открытие файла

        // сохранение данных
        for (int i = 0; i < 10; i++)
            fwrite(&recs[i], sizeof(srec), 1, f);

        fclose(f);
    }

// отображение таблицы рекордов
void CGame::drawrec() {
    disp.col(15, 0);
    disp.clear();
    disp.col(15, 0);
    disp.pos(21, 3);
    printf("Таблица рекордов");
    disp.col(15, 0);
    disp.pos(18, 6);
    printf("Счет\n");
    disp.col(15, 0);
    // вывод строк таблицы
    const char* format1 = "%2d %-20d";
    const char* format2 = "%2d -";
    for (int i = 0; i < 10; i++) {
        disp.pos(13, 7 + i);
        int och = recs[i].ochki;
        int s = i;
        _asm {
            mov eax, och // в eax переменную
            cmp eax, 0 // сравниваем och с 0
            jle ELSE // переход к метке ELSE, если меньше или равно
            mov ebx, och // в ebx переменную och
            push ebx // закидываем в стек
            mov eax, s // в ebx переменную s
            inc eax // прибавляем 1
            push eax // закидываем в стек
        }
    }
}

```

```

push format1 // закидываем в стек
call printf // вызываем функцию printf
add esp, 12 // повышаем вершину стека на 12
jmp ENDIF

ELSE:

mov ebx, s // в ebx переменную s
inc ebx // прибавляем 1
push ebx // закидываем в стек
push format2 // закидываем в стек
call printf // вызываем функцию
add esp, 8 // повышаем вершину стека на 8

ENDIF:
}

}
_getch();
}

```

// запуск главного меню и выполнение действий по пунктам меню

```
void CGame::run()
```

```

{
    int k = 0;
    readrec();
    while (1)
    {
        k = menu(k); // выполнение меню
        if (k == 2)
            return;
        // выполнение пункта меню
        switch (k)
        {
            case 0: game(true);
                    break;
            case 1: drawrec();

```

```

        break;
    case 2: return; // выход
    }
}

```

Приложение В Схемы программы.

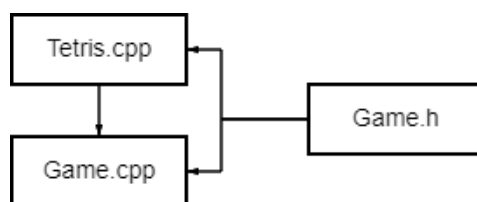


Рисунок 5 — взаимодействие программы

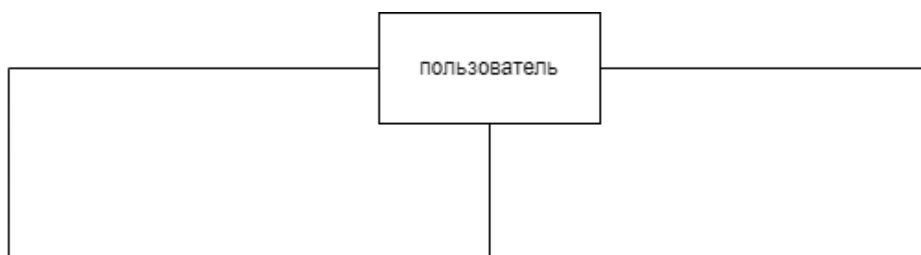


Рисунок 6 — схема данных

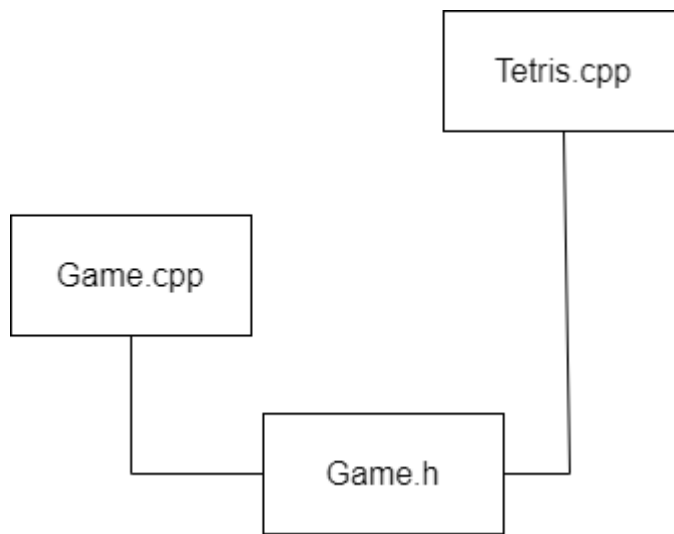


Рисунок 7 — иерархическая схема программы



Рисунок 8 — функция main

Схема программы. Функция run

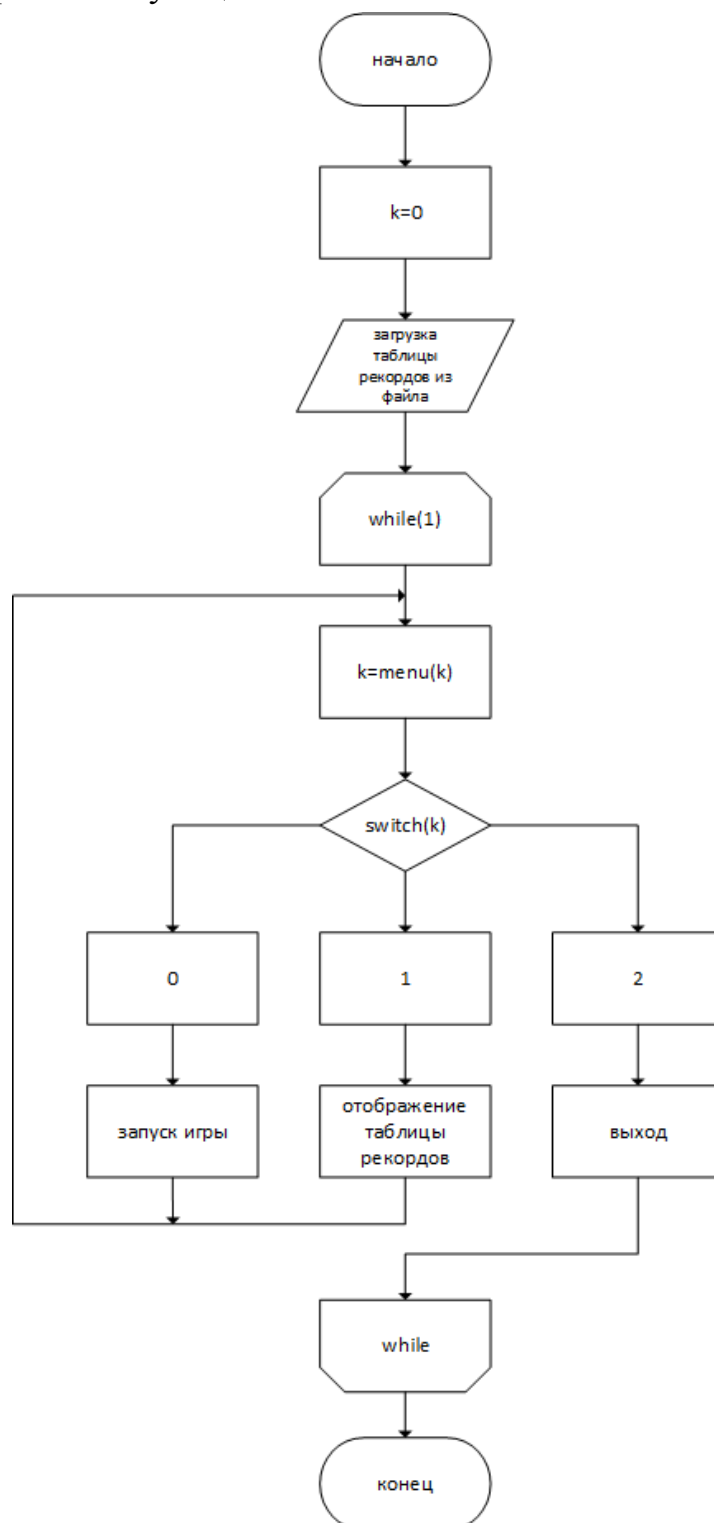


Рисунок 9 — функция `run`: запуск главного меню и выполнение действий по пунктам меню.

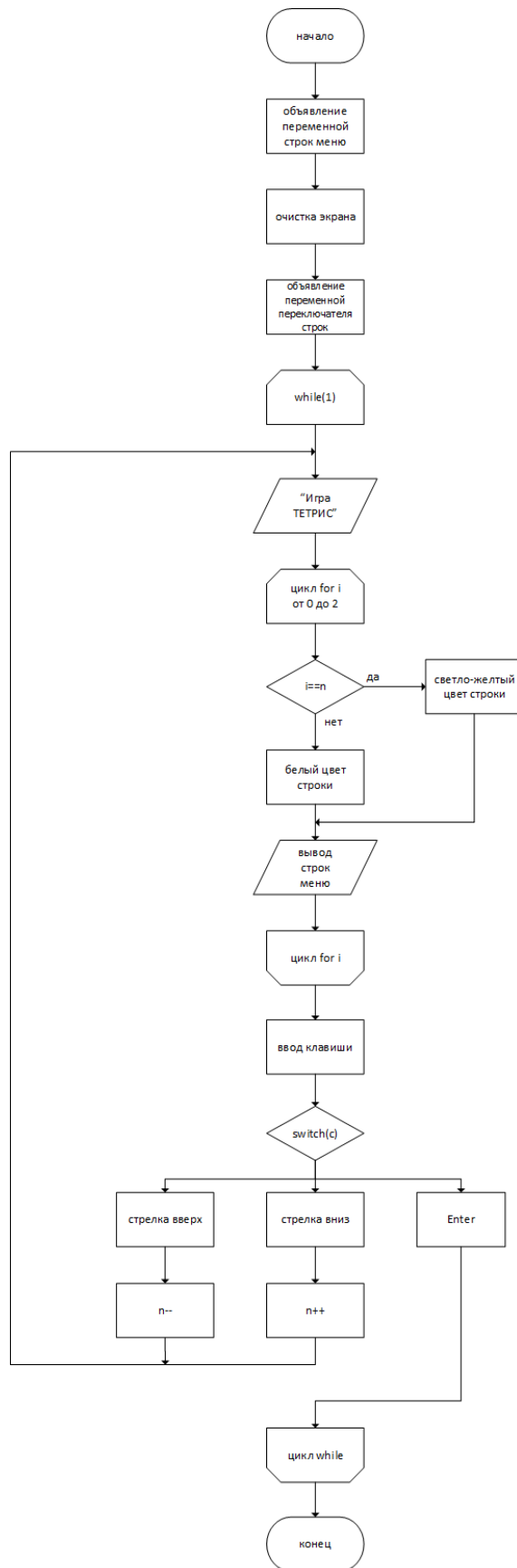


Рисунок 10 — функция теги: отображение меню

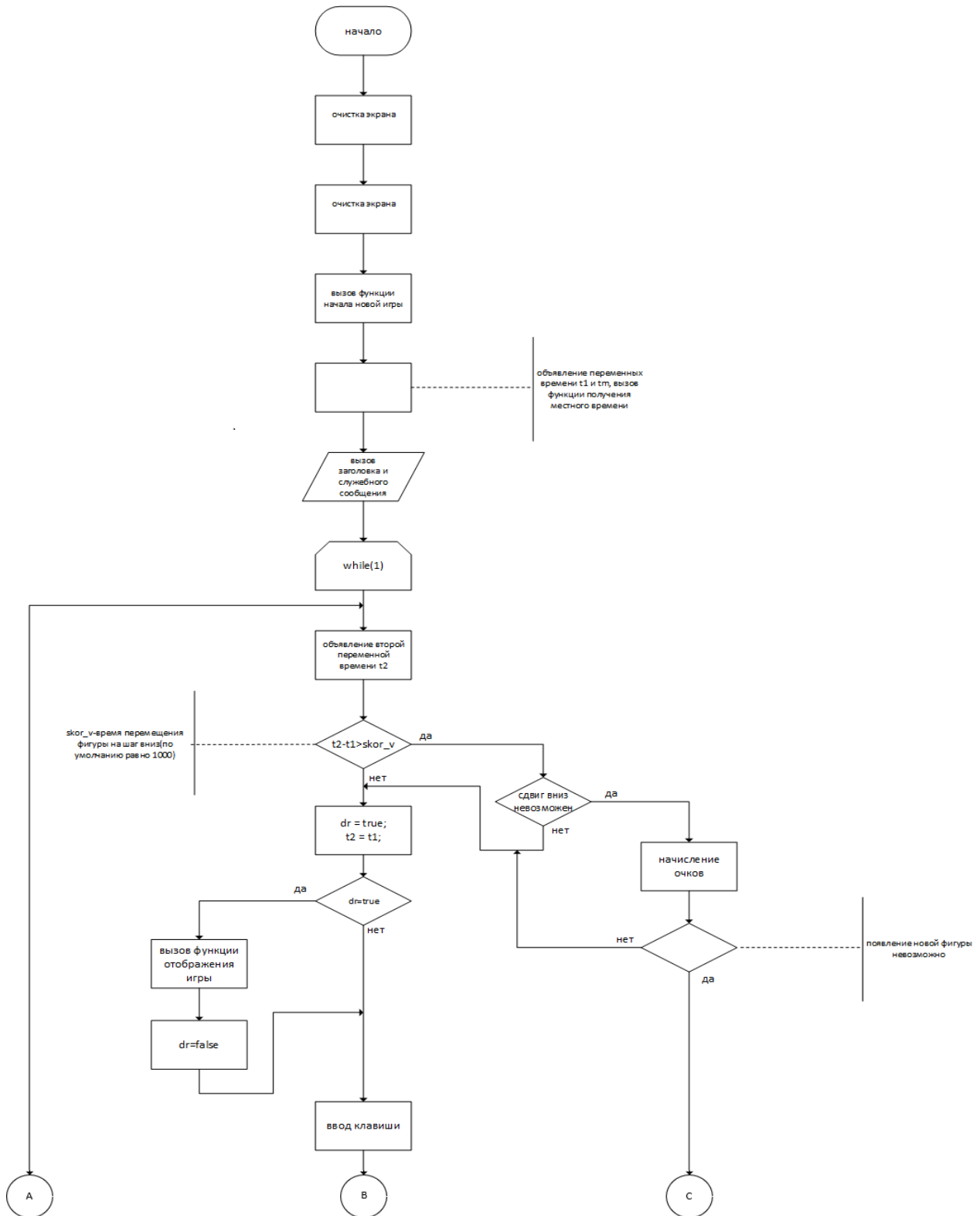


Рисунок 11 - Функция game

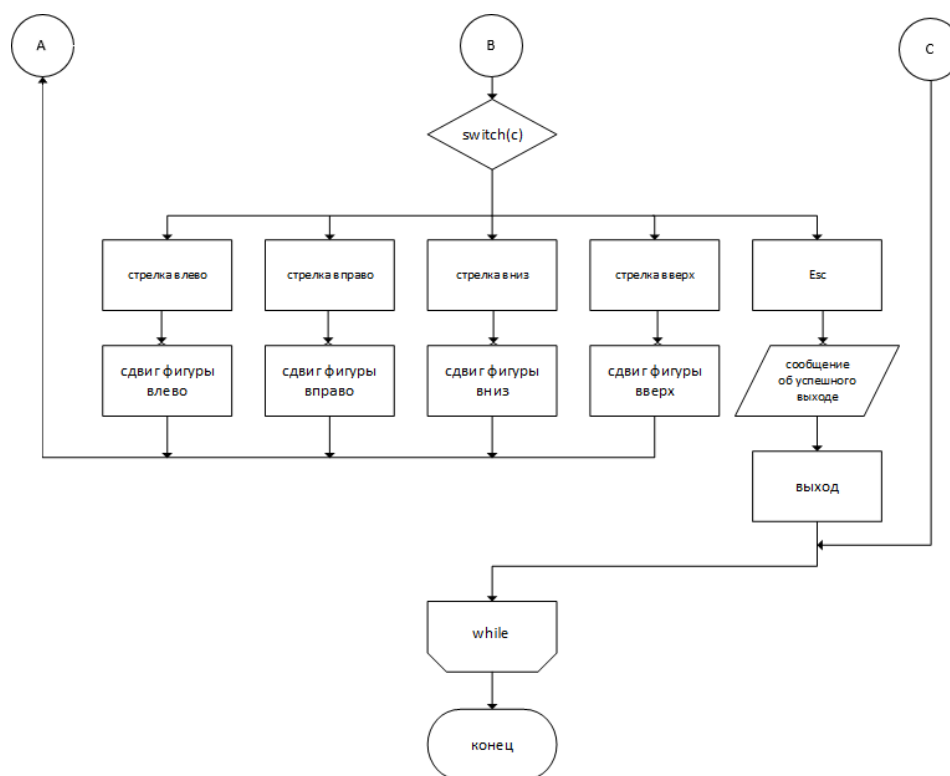


Рисунок 12 — функция game:запуск игры

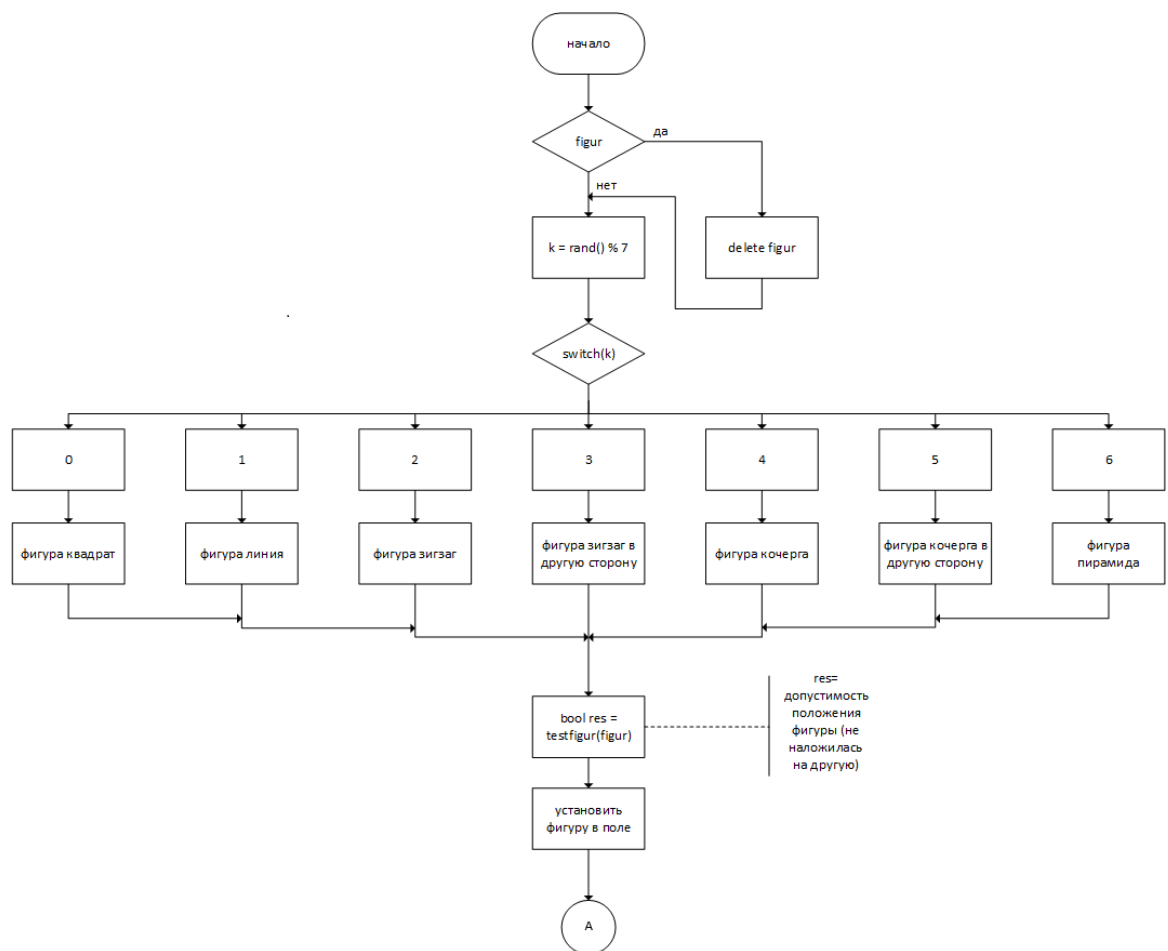


Рисунок 13 — функция newfigur

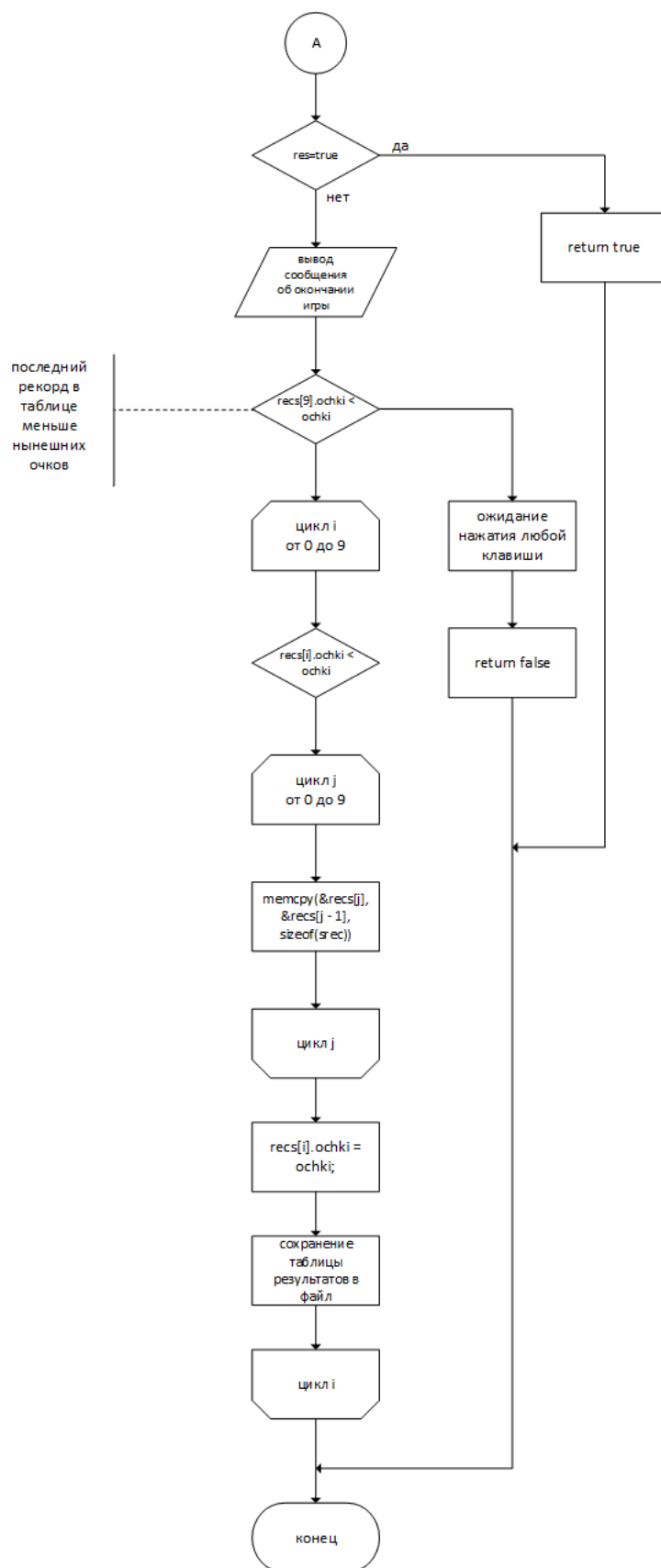


Рисунок 14 — функция newfigur:случайный выбор следующей фигуры

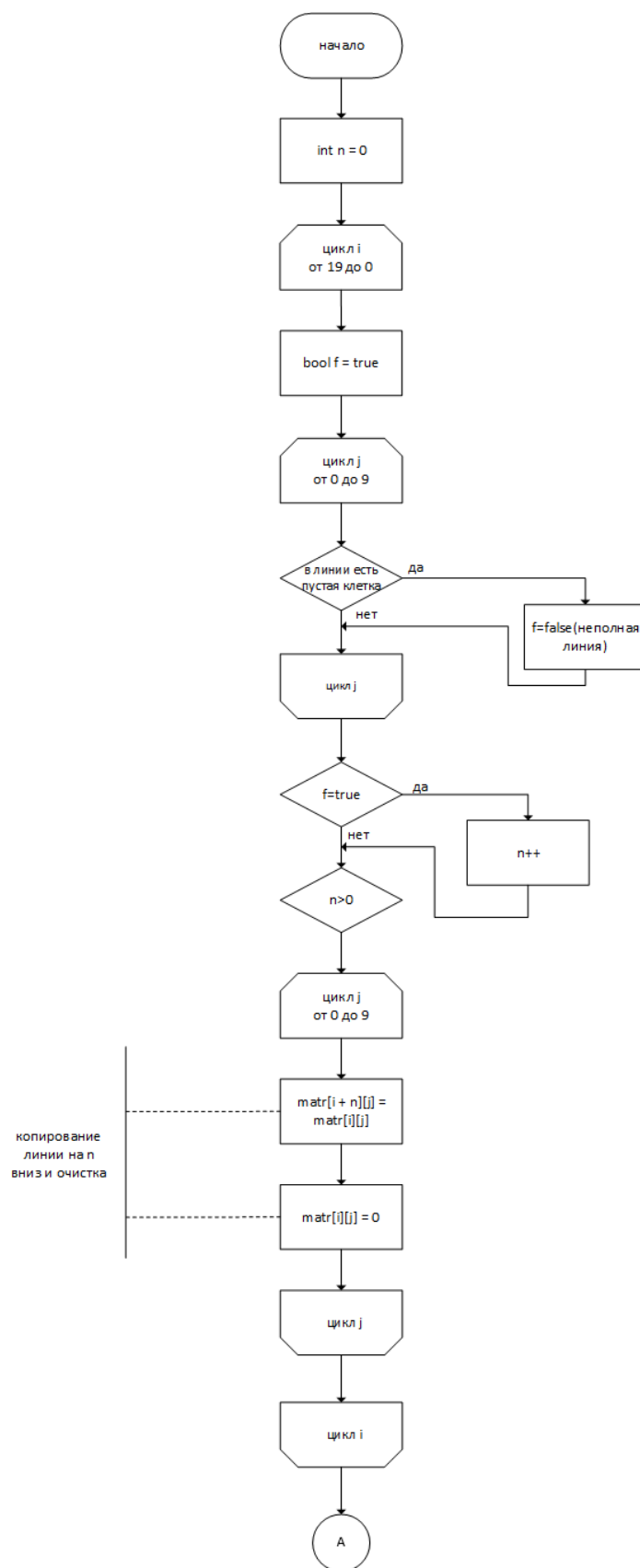


Рисунок 15 – функция nachisl

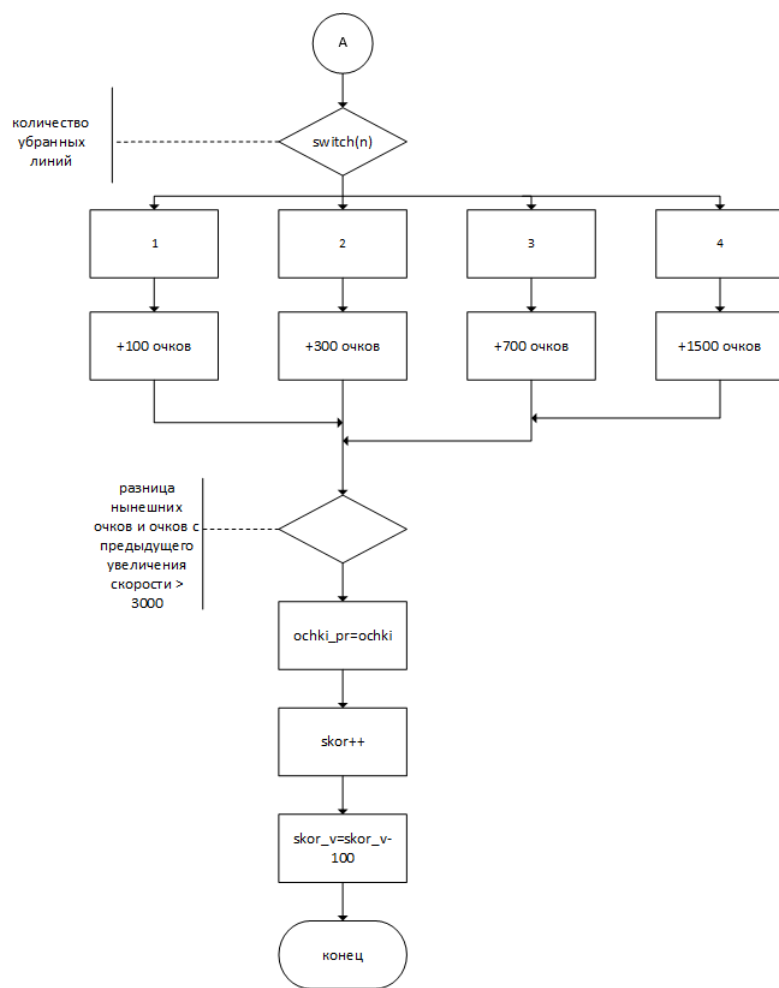


Рисунок 16— функция nachisl: удаление линий, если они заполнены и начисление очков

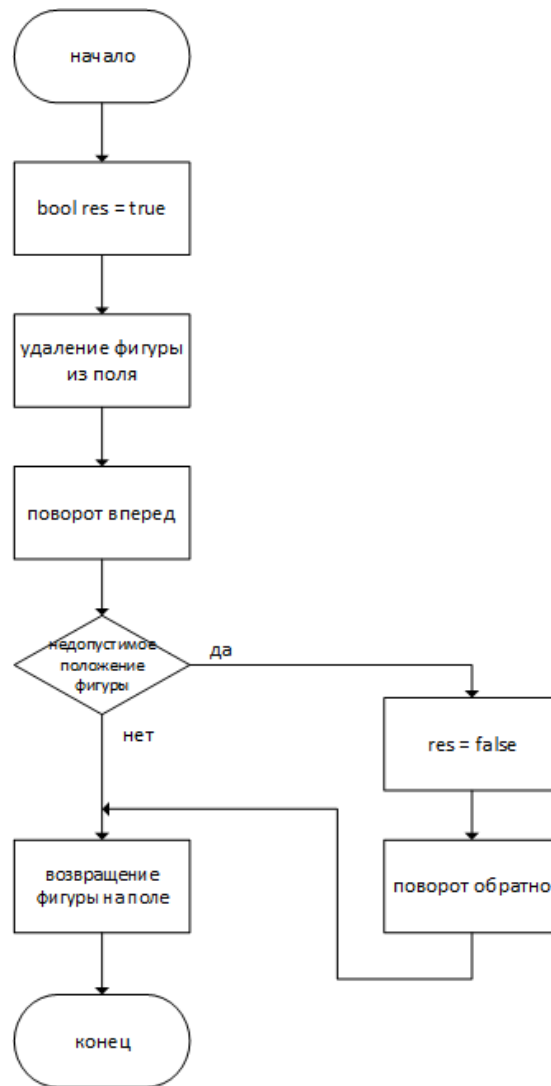


Рисунок 17 — функция `rotate`: поворот фигуры

Схема программы. Функция sdvig

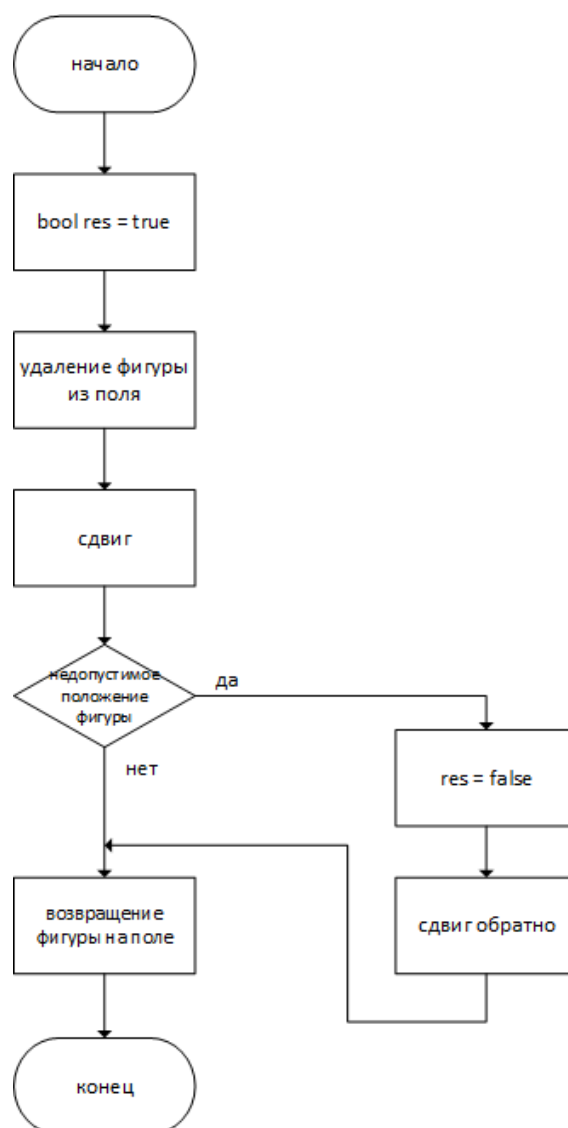


Рисунок 18— функция `sdvig`:сдвиг фигуры

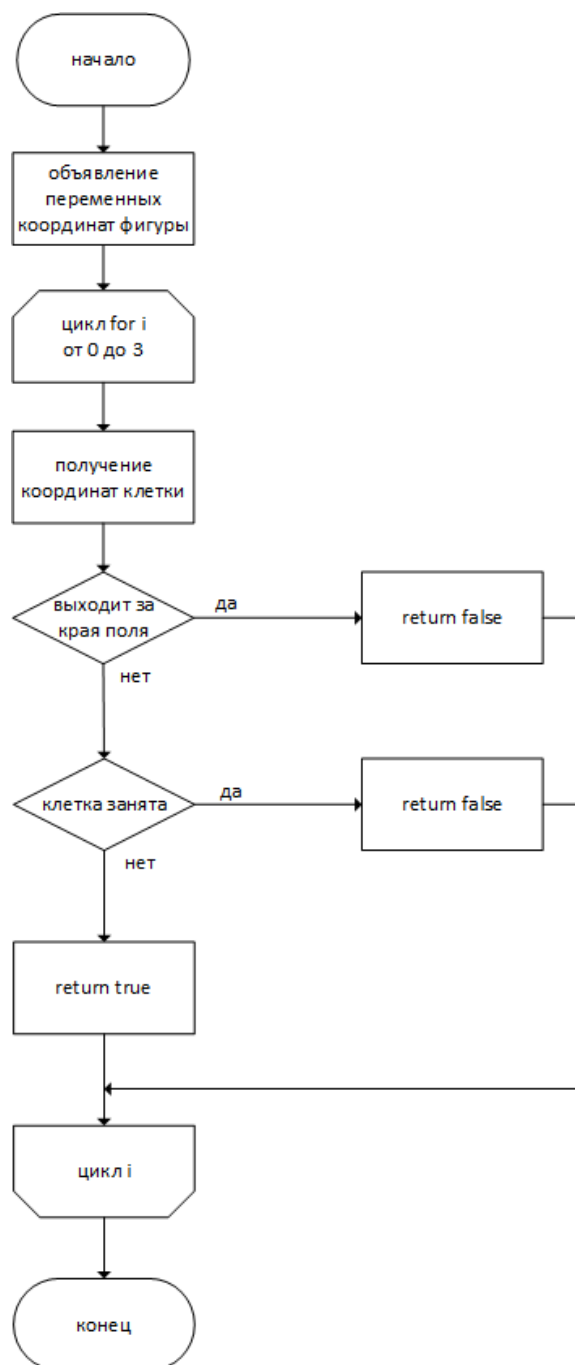


Рисунок 19 — функция `testfigur`: проверка возможности размещения фигуры



Рисунок 20 — функция addfigur: размещение фигуры в поле

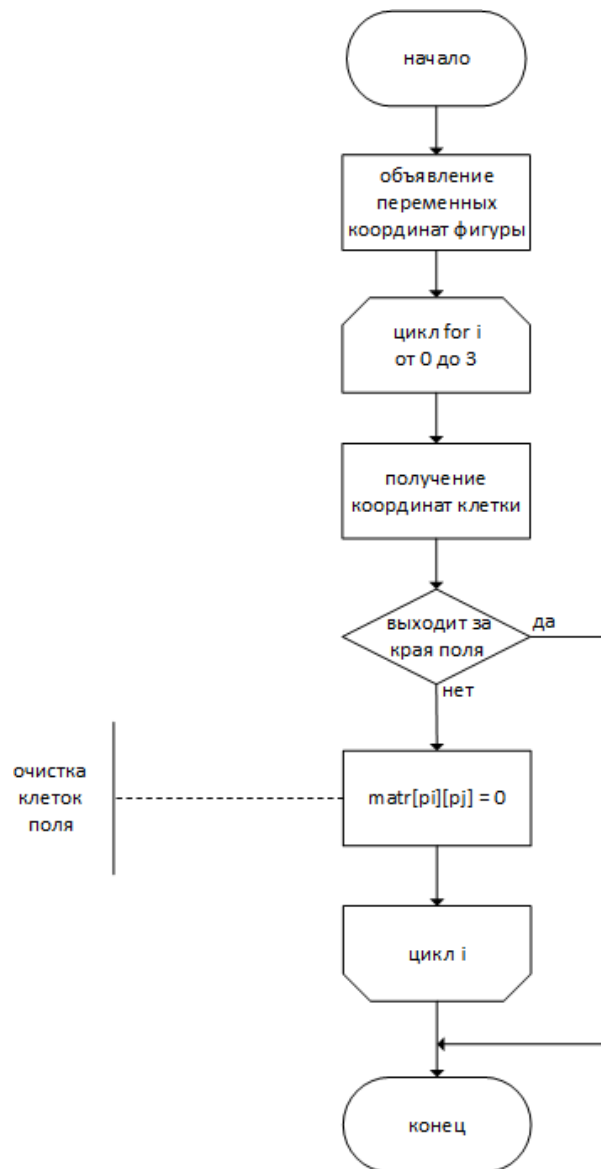


Рисунок 21 — функция deletefigur: удаление фигуры из поля