Федеральное агенство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования

"Сибирский государственный университет телекоммуникаций и информатики"

Курсовая работа

по дисциплине "Программирование"

ВАРИАНТ 20

Выполнил: студент группы ИП - 913

\*

Проверил: ассистент кафедры ПМиК

Дьячкова И.С.

Новосибирск 2020

**Содержание:**

1. **Введение**
2. **Описание алгоритма**
3. **Код программы на C++**
4. **Результат работы программы**

**1. Введение**

**Задача:**

Выполнить Курсовую работу по дисциплине "Программирование" за 2 семестр

**Описание:**

Согласно заданию для моего варианта необходимо: «Написать программу для игры Тетрис».

**2. Описание алгоритма**

**Определения**

Стартовые координаты, используемые для отрисовки поля, интерфейса и тетромино

#define XSTARTCORD 60

#define YSTARTCORD 20

#define XINTERFACE 420

#define YINTERFACE 20

#define XTETROMINOSTARTCORD 20

#define YTETROMINOSTARTCORD 60

Количество колонок и строк в поле

#define ROWS 20

#define COLS 15

#define BLOCKSIZE 20 - сторона кубика

Цветовая палитра

#define BLACK 0

#define BLUE 1

#define GREEN 2

#define CYAN 3

#define RED 4

#define PURPLE 5

#define BROWN 6

#define LIGHTGRAY 7

#define DARKGRAY 8

#define LIGHTBLUE 9

#define LIGHTGREEN 10

#define LIGHTCYAN 11

#define LIGHTRED 12

#define LIGHTPURPLE 13

#define YELLOW 14

#define WHITE 15

Список нажимаемых клавиш

#define ESC 27

#define UP 72

#define LEFT 75

#define RIGHT 77

#define DOWN 80

#define ENTER 13

**Функции**

void field\_init(); - заполняет массив игрового поля начальными нулями

int tetromino\_blocks\_init(); - выделяет память и сохраняет изображения блоков разных цветов в буферы для дальнейшего использования

void generate\_tetromino(int tetromino[][4], int \*color); - генерирует новое тетромино с случайными цветом и типом

void draw\_field(); - отрисовывет игровое поле для тетриса

void draw\_interface(); - отрисовывает интерфейс

void enter\_screen(); - отрисовывает экран для начала игры

void draw\_tetromino\_block(int X, int Y, int Color); - отрисовывает по одному блоку за вызов

void draw\_tetromino(int X\_current, int Y\_current, int Color); - вызывает отрисовку блоков, чтоб отрисовать целое тетромино

void Refresh\_field(); - обновляет текущее состояние игрового поля

void DeleteLines(); - удаляет заполненные полоски в игровом поле

int move\_tetromino(int direction); - двигает тетромино в указанную сторону

int border\_bump(int direction); - проверяет тетромино на столкновение с границей поля

int tetromino\_bump(int direction); - проверяет тетромино на столкновение с другими

int pin\_tetromino(); - закрепляет тетромино на игровом поле после столкновения с нижней границей

void take\_new\_tetromino(); - вызывает генерацию нового тетромино для продолжения игры

**Переменные**

Матрицы с расположениями блоков в конструкции разных типов тетромин

const int I\_tetromino[4][4];

const int I\_tetromino\_turned[4][4];

const int O\_tetromino[4][4];

const int T\_tetromino[4][4];

const int T\_tetromino\_turned[4][4];

const int L\_tetromino[4][4];

const int L\_tetromino\_turned[4][4];

const int L2\_tetromino[4][4];

const int L2\_tetromino\_turned[4][4];

const int Z\_tetromino[4][4];

const int Z\_tetromino\_turned[4][4];

const int Z2\_tetromino[4][4];

const int Z\_tetromino\_turned[4][4];

Буферы для запоминания изображения блоков разных цветов

void far \*BlackBlockBuf;

void far \*GreenBlockBuf;

void far \*CyanBlockBuf;

void far \*RedBlockBuf;

void far \*PurpleBlockBuf;

void far \*YellowBlockBuf;

int tetris\_field[ROWS][COLS]; - матрица состояния игрового поля

int current\_tetromino[4][4]; - матрица состояния текущего тетромино

int current\_tetromino\_index; - индекс текущего типа тетромино

int current\_color\_index; - индекс текущего цвета тетромино

int next\_tetromino[4][4]; - матрица состояния следующего тетромино

int next\_tetromino\_index; - индекс типа следующего тетромино

int next\_color\_index; - индекс следующего цвета тетромино

int turn\_index; - показатель разворота тетромино

int move\_time\_limit; - лимит таймера для хода вниз

int move\_time\_passed; - счётчик таймера

Стартовые координаты тетромино в матрице состояния

const int X\_tetromino\_start;

const int Y\_tetromino\_start;

Текущие координаты тетромино в матрице состояния

int current\_tetromino\_X;

int current\_tetromino\_Y;

int temp\_tetromino[4][4]; - матрица временного состояния тетромино для проверки сталкивания с другими

int temp\_turn\_index; - временный индекс показателья разворота

int score; - счётчик игрового счёта

3**. Код программы на C++**

#include <iostream>

#include <math.h>

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctime>

using namespace std;

#define XSTARTCORD 60

#define YSTARTCORD 20

#define XINTERFACE 420

#define YINTERFACE 20

#define XTETROMINOSTARTCORD 20

#define YTETROMINOSTARTCORD 60

#define ROWS 20

#define COLS 15

#define BLOCKSIZE 20

#define BLACK 0

#define BLUE 1

#define GREEN 2

#define CYAN 3

#define RED 4

#define PURPLE 5

#define BROWN 6

#define LIGHTGRAY 7

#define DARKGRAY 8

#define LIGHTBLUE 9

#define LIGHTGREEN 10

#define LIGHTCYAN 11

#define LIGHTRED 12

#define LIGHTPURPLE 13

#define YELLOW 14

#define WHITE 15

#define ESC 27

#define UP 72

#define LEFT 75

#define RIGHT 77

#define DOWN 80

#define ENTER 13

void field\_init();

int tetromino\_blocks\_init();

void generate\_tetromino(int tetromino[][4], int \*color);

void draw\_field();

void draw\_interface();

void enter\_screen();

void draw\_tetromino(int X\_current, int Y\_current, int Color);

void draw\_tetromino\_block(int X, int Y, int Color);

void Refresh\_field();

void DeleteLines();

int move\_tetromino(int direction);

int border\_bump(int direction);

int tetromino\_bump(int direction);

int pin\_tetromino();

void take\_new\_tetromino();

const int I\_tetromino[4][4] = { 0,1,0,0, 0,1,0,0, 0,1,0,0, 0,1,0,0};

const int I\_tetromino\_turned[4][4] = { 0,0,0,0, 1,1,1,1, 0,0,0,0, 0,0,0,0};

const int O\_tetromino[4][4] = { 1,1,0,0, 1,1,0,0, 0,0,0,0, 0,0,0,0};

const int T\_tetromino[4][4] = { 0,1,0,0, 0,1,1,0, 0,1,0,0, 0,0,0,0};

const int T\_tetromino\_turned1[4][4] = { 0,1,0,0, 1,1,1,0, 0,0,0,0, 0,0,0,0};

const int T\_tetromino\_turned2[4][4] = {0,1,0,0, 1,1,0,0, 0,1,0,0, 0,0,0,0};

const int T\_tetromino\_turned3[4][4] = { 0,0,0,0,1,1,1,0, 0,1,0,0, 0,0,0,0};

const int L\_tetromino[4][4] = { 0,1,0,0, 0,1,0,0, 0,1,1,0, 0,0,0,0};

const int L\_tetromino\_turned1[4][4] = { 0,0,1,0, 1,1,1,0, 0,0,0,0, 0,0,0,0 };

const int L\_tetromino\_turned2[4][4] = { 1,1,0,0, 0,1,0,0, 0,1,0,0, 0,0,0,0 };

const int L\_tetromino\_turned3[4][4] = {0,0,0,0, 1,1,1,0, 1,0,0,0, 0,0,0,0};

const int L2\_tetromino[4][4] = { 0,1,0,0, 0,1,0,0, 1,1,0,0, 0,0,0,0 };

const int L2\_tetromino\_turned1[4][4] = { 0,0,0,0, 1,1,1,0, 0,0,1,0, 0,0,0,0};

const int L2\_tetromino\_turned2[4][4] = { 0,1,1,0, 0,1,0,0, 0,1,0,0, 0,0,0,0};

const int L2\_tetromino\_turned3[4][4] = { 1,0,0,0, 1,1,1,0, 0,0,0,0, 0,0,0,0};

const int Z\_tetromino[4][4] = { 0,0,0,0, 0,1,1,0, 1,1,0,0, 0,0,0,0};

const int Z\_tetromino\_turned[4][4] = { 1,0,0,0, 1,1,0,0, 0,1,0,0, 0,0,0,0};

const int Z2\_tetromino[4][4] = { 0,0,0,0, 1,1,0,0, 0,1,1,0, 0,0,0,0 };

const int Z2\_tetromino\_turned[4][4] = { 0,1,0,0, 1,1,0,0, 1,0,0,0, 0,0,0,0 };

void far \*BlackBlockBuf;

void far \*GreenBlockBuf;

void far \*CyanBlockBuf;

void far \*RedBlockBuf;

void far \*PurpleBlockBuf;

void far \*YellowBlockBuf;

int tetris\_field[ROWS][COLS];

int current\_tetromino[4][4];

int current\_tetromino\_index = -1;

int current\_color\_index = BLACK;

int next\_tetromino[4][4];

int next\_tetromino\_index = -1;

int next\_color\_index = BLACK;

int turn\_index = 0;

int move\_time\_limit = 80;

int move\_time\_passed = 0;

const int X\_tetromino\_start = 0;

const int Y\_tetromino\_start = 6;

int current\_tetromino\_X = X\_tetromino\_start;

int current\_tetromino\_Y = Y\_tetromino\_start;

int temp\_tetromino[4][4];

int temp\_turn\_index = 0;

int score = 0;

void draw\_field(){

cleardevice(); // очистка окна

setcolor(BLUE);

moveto(XSTARTCORD-3, YSTARTCORD-3);

lineto(XSTARTCORD+301,YSTARTCORD-3);

lineto(XSTARTCORD+301,YSTARTCORD+401);

lineto(XSTARTCORD-3,YSTARTCORD+401);

lineto(XSTARTCORD-3,YSTARTCORD-3);

return;

}

void draw\_interface(){

setcolor(BROWN);

moveto(XINTERFACE, YINTERFACE);

lineto(XINTERFACE+200,YINTERFACE);

lineto(XINTERFACE+200,YINTERFACE+230);

lineto(XINTERFACE,YINTERFACE+230);

lineto(XINTERFACE,YINTERFACE);

setcolor(WHITE);

settextstyle(0,0,1);

outtextxy(XINTERFACE+60,YINTERFACE+10, "Управление:");

outtextxy(XINTERFACE+15,YINTERFACE+45, " ^");

outtextxy(XINTERFACE+15,YINTERFACE+50, "Повернуть: |");

outtextxy(XINTERFACE+15,YINTERFACE+80, "Влево: <--");

outtextxy(XINTERFACE+15,YINTERFACE+110, "Вправо: -->");

outtextxy(XINTERFACE+15,YINTERFACE+153, " v");

outtextxy(XINTERFACE+15,YINTERFACE+140, "Вниз: |");

outtextxy(XINTERFACE+15,YINTERFACE+180, "Пауза: Enter");

outtextxy(XINTERFACE+15,YINTERFACE+200, "Выйти: ESC");

setcolor(DARKGRAY);

moveto(XINTERFACE, YINTERFACE+250);

lineto(XINTERFACE+200,YINTERFACE+250);

lineto(XINTERFACE+200,YINTERFACE+430);

lineto(XINTERFACE,YINTERFACE+430);

lineto(XINTERFACE,YINTERFACE+250);

setcolor(WHITE);

outtextxy(XINTERFACE+25,YINTERFACE+260, "Следующая фигура:");

return;

}

void enter\_screen(){

moveto(XSTARTCORD+20, YSTARTCORD+150);

setcolor(LIGHTRED);

lineto(XSTARTCORD+280,YSTARTCORD+150);

lineto(XSTARTCORD+280,YSTARTCORD+250);

lineto(XSTARTCORD+20,YSTARTCORD+250);

lineto(XSTARTCORD+20,YSTARTCORD+150);

setcolor(WHITE);

settextstyle(10,0,2);

outtextxy(XSTARTCORD+110,YSTARTCORD+180, "Tetris");

settextstyle(0,0,1);

outtextxy(XSTARTCORD+30,YSTARTCORD+220, "Нажмите ENTER чтобы начать");

char KEY;

while(1){

if(kbhit()){

KEY = getch();

if(KEY == ENTER){

break;

}

}

}

setfillstyle ( 1, BLACK );

bar(XSTARTCORD+20, YSTARTCORD+150, XSTARTCORD+290, YSTARTCORD+260);

setcolor(WHITE);

outtextxy(XSTARTCORD+15,YSTARTCORD+430, "Нажмите ENTER чтобы остановить");

outtextxy(XSTARTCORD,YSTARTCORD+470, "Cчёт: 0");

return;

}

void field\_init(){

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

for (int j=0; j<COLS;j++){

tetris\_field[i][j]=0;

}

}

return;

}

int tetromino\_blocks\_init(){

unsigned int size;

//рисуем заготовки кубиков для тетромино

setfillstyle ( 1, BLACK );

bar(10, 10, 10+BLOCKSIZE, 10+BLOCKSIZE);

setfillstyle ( 1, GREEN );

bar(40, 10, 40+BLOCKSIZE, 10+BLOCKSIZE);

setfillstyle ( 1, LIGHTGREEN );

bar(45, 15, 35+BLOCKSIZE, 5+BLOCKSIZE);

setfillstyle ( 1, CYAN );

bar(70, 10, 70+BLOCKSIZE, 10+BLOCKSIZE);

setfillstyle ( 1, LIGHTCYAN );

bar(75, 15, 65+BLOCKSIZE, 5+BLOCKSIZE);

setfillstyle ( 1, RED );

bar(100, 10, 100+BLOCKSIZE, 10+BLOCKSIZE);

setfillstyle ( 1, LIGHTRED );

bar(105, 15, 95+BLOCKSIZE, 5+BLOCKSIZE);

setfillstyle ( 1, PURPLE );

bar(130, 10, 130+BLOCKSIZE, 10+BLOCKSIZE);

setfillstyle ( 1, LIGHTPURPLE );

bar(135, 15, 125+BLOCKSIZE, 5+BLOCKSIZE);

setfillstyle ( 1, BROWN );

bar(160, 10, 160+BLOCKSIZE, 10+BLOCKSIZE);

setfillstyle ( 1, YELLOW );

bar(165, 15, 155+BLOCKSIZE, 5+BLOCKSIZE);

size= imagesize(10, 10, 10+BLOCKSIZE, 10+BLOCKSIZE);

if (size != 0xffff) {

BlackBlockBuf = malloc(size);

if(BlackBlockBuf) {

getimage(10, 10, 10+BLOCKSIZE, 10+BLOCKSIZE, BlackBlockBuf);

}

else

return 0;

}

size= imagesize(40, 10, 40+BLOCKSIZE, 10+BLOCKSIZE);

if (size != 0xffff) {

GreenBlockBuf= malloc(size);

if(GreenBlockBuf) {

getimage(40, 10, 40+BLOCKSIZE, 10+BLOCKSIZE, GreenBlockBuf);

}

else

return 0;

}

size= imagesize(70, 10, 70+BLOCKSIZE, 10+BLOCKSIZE);

if (size != 0xffff) {

CyanBlockBuf= malloc(size);

if(CyanBlockBuf) {

getimage(70, 10, 70+BLOCKSIZE, 10+BLOCKSIZE, CyanBlockBuf);

}

else

return 0;

}

size= imagesize(100, 10, 100+BLOCKSIZE, 10+BLOCKSIZE);

if (size != 0xffff) {

RedBlockBuf= malloc(size);

if(RedBlockBuf) {

getimage(100, 10, 100+BLOCKSIZE, 10+BLOCKSIZE, RedBlockBuf);

}

else

return 0;

}

size= imagesize(130, 10, 130+BLOCKSIZE, 10+BLOCKSIZE);

if (size != 0xffff) {

PurpleBlockBuf= malloc(size);

if(PurpleBlockBuf) {

getimage(130, 10, 130+BLOCKSIZE, 10+BLOCKSIZE, PurpleBlockBuf);

}

else

return 0;

}

size= imagesize(160, 10, 160+BLOCKSIZE, 10+BLOCKSIZE);

if (size != 0xffff) {

YellowBlockBuf= malloc(size);

if(YellowBlockBuf) {

getimage(160, 10, 160+BLOCKSIZE, 10+BLOCKSIZE, YellowBlockBuf);

}

else

return 0;

}

return 1;

}

void generate\_tetromino(int tetromino[][4], int \*color, int \*tetromino\_index){

int r = rand() % 7;

switch(r) {

case 0 :

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

tetromino[i][j] = I\_tetromino[i][j];

}

}

\*tetromino\_index = 0;

break;

case 1 :

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

tetromino[i][j] = O\_tetromino[i][j];

}

}

\*tetromino\_index = 1;

break;

case 2 :

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

tetromino[i][j] = T\_tetromino[i][j];

}

}

\*tetromino\_index = 2;

break;

case 3 :

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

tetromino[i][j] = L\_tetromino[i][j];

}

}

\*tetromino\_index = 3;

break;

case 4 :

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

tetromino[i][j] = L2\_tetromino[i][j];

}

}

\*tetromino\_index = 4;

break;

case 5 :

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

tetromino[i][j] = Z\_tetromino[i][j];

}

}

\*tetromino\_index = 5;

break;

case 6 :

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

tetromino[i][j] = Z2\_tetromino[i][j];

}

}

\*tetromino\_index = 6;

break;

}

r = rand() % 5;

switch(r){

case 0 :

\*color = GREEN;

break;

case 1:

\*color = CYAN;

break;

case 2:

\*color = RED;

break;

case 3:

\*color = PURPLE;

break;

case 4:

\*color = YELLOW;

break;

}

return;

}

void draw\_tetromino(int X\_current, int Y\_current, int Color){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

if(current\_tetromino[i][j]==1){

draw\_tetromino\_block(X\_current+i\*BLOCKSIZE,Y\_current+j\*BLOCKSIZE,Color);

}

}

}

return;

}

void draw\_tetromino\_block(int X, int Y, int Color){

switch(Color){

case BLACK:

putimage(Y, X, BlackBlockBuf, COPY\_PUT);

break;

case GREEN:

putimage(Y, X, GreenBlockBuf, COPY\_PUT);

break;

case CYAN:

putimage(Y, X, CyanBlockBuf, COPY\_PUT);

break;

case RED:

putimage(Y, X, RedBlockBuf, COPY\_PUT);

break;

case PURPLE:

putimage(Y, X, PurpleBlockBuf, COPY\_PUT);

break;

case YELLOW:

putimage(Y, X, YellowBlockBuf, COPY\_PUT);

break;

default:

cout << "Неизвестный цвет"<<endl;

break;

}

return;

}

void turn\_tetromino(){

switch(current\_tetromino\_index){

case 0:

if(turn\_index == 0){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = I\_tetromino\_turned[i][j];

}

}

turn\_index = 1;

}

else{

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = I\_tetromino[i][j];

}

}

turn\_index = 0;

}

break;

case 1:

break;

case 2:

if(turn\_index == 0){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = T\_tetromino\_turned1[i][j];

}

}

turn\_index = 1;

}

else{

if(turn\_index == 1){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = T\_tetromino\_turned2[i][j];

}

}

turn\_index = 2;

}

else{

if(turn\_index == 2){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = T\_tetromino\_turned3[i][j];

}

}

turn\_index = 3;

}

else{

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = T\_tetromino[i][j];

}

}

turn\_index = 0;

}

}

}

break;

case 3:

if(turn\_index == 0){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L\_tetromino\_turned1[i][j];

}

}

turn\_index = 1;

}

else{

if(turn\_index == 1){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L\_tetromino\_turned2[i][j];

}

}

turn\_index = 2;

}

else{

if(turn\_index == 2){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L\_tetromino\_turned3[i][j];

}

}

turn\_index = 3;

}

else{

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L\_tetromino[i][j];

}

}

turn\_index = 0;

}

}

}

break;

case 4:

if(turn\_index == 0){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L2\_tetromino\_turned1[i][j];

}

}

turn\_index = 1;

}

else{

if(turn\_index == 1){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L2\_tetromino\_turned2[i][j];

}

}

turn\_index = 2;

}

else{

if(turn\_index == 2){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L2\_tetromino\_turned3[i][j];

}

}

turn\_index = 3;

}

else{

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = L2\_tetromino[i][j];

}

}

turn\_index = 0;

}

}

}

break;

case 5:

if(turn\_index == 0){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = Z\_tetromino\_turned[i][j];

}

}

turn\_index = 1;

}

else{

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = Z\_tetromino[i][j];

}

}

turn\_index = 0;

}

break;

case 6:

if(turn\_index == 0){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = Z2\_tetromino\_turned[i][j];

}

}

turn\_index = 1;

}

else{

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = Z2\_tetromino[i][j];

}

}

turn\_index = 0;

}

break;

}

return;

}

void Refresh\_field(){

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

for (int j=0; j<COLS;j++){

draw\_tetromino\_block(XTETROMINOSTARTCORD+i\*BLOCKSIZE, YTETROMINOSTARTCORD+j\*BLOCKSIZE, BLACK);

}

}

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

for (int j=0; j<COLS;j++){

if(tetris\_field[i][j]!=0){

draw\_tetromino\_block(XTETROMINOSTARTCORD+i\*BLOCKSIZE, YTETROMINOSTARTCORD+j\*BLOCKSIZE, tetris\_field[i][j]);

}

}

}

for (int i=0;i<4;i++){

for (int j=0; j<4;j++){

draw\_tetromino\_block(YINTERFACE+320+i\*BLOCKSIZE, XINTERFACE+72+j\*BLOCKSIZE, BLACK);

}

}

for (int i=0;i<4;i++){

for (int j=0; j<4;j++){

if(next\_tetromino[i][j]!=0){

draw\_tetromino\_block(YINTERFACE+320+i\*BLOCKSIZE, XINTERFACE+72+j\*BLOCKSIZE, next\_color\_index);

}

}

}

return;

}

void DeleteLines(){

int line\_counter=0;

for (int i=1;i<ROWS;i++){

line\_counter=0;

for (int j=0; j<COLS;j++){

if(tetris\_field[i][j]!=0){

line\_counter++;

}

}

if(line\_counter==15){

for(int v=i; v>0;v--){

for (int j=0; j<COLS;j++){

tetris\_field[v][j]=tetris\_field[v-1][j];

}

score+=150;

}

}

}

}

int move\_tetromino(int direction){

switch(direction){

case DOWN:

if(border\_bump(DOWN)==1){

if(tetromino\_bump(DOWN)==1){

draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE,BLACK);

current\_tetromino\_X++;

return 0;

}

}

return 1;

case UP:

if(border\_bump(UP)==1){

if(tetromino\_bump(UP)==1){ draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE,BLACK);

turn\_tetromino(); draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE, current\_color\_index);

}

}

return 0;

case LEFT:

if(border\_bump(LEFT)==1){

if(tetromino\_bump(LEFT)==1){ draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE,BLACK);

current\_tetromino\_Y--; draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE, current\_color\_index);

}

}

return 0;

case RIGHT:

if(border\_bump(RIGHT)==1) {

if(tetromino\_bump(RIGHT)==1){

draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE,BLACK);

current\_tetromino\_Y++; draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE, current\_color\_index);

}

}

return 0;

}

}

int border\_bump(int direction){

switch(direction){

case DOWN:

if((current\_tetromino\_X>=16) && (current\_tetromino[3][1]==1)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_X>=17) && ((current\_tetromino[2][0]==1) || (current\_tetromino[2][1]==1) || (current\_tetromino[2][2]==1))){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_X>=18) && (current\_tetromino[1][1]==1)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

case UP:

switch(current\_tetromino\_index){

case 0:

if((current\_tetromino\_X>=17) && (current\_tetromino[1][0]==1)){

cout << "Невохможно повернуться"<<endl;

return 0;

}

if((current\_tetromino\_Y<0) && (current\_tetromino[0][1]==1)){

cout << "Невохможно повернуться"<<endl;

return 0;

}

if((current\_tetromino\_Y>11) && (current\_tetromino[0][1]==1)){

cout << "Невохможно повернуться"<<endl;

return 0;

}

return 1;

case 1:

return 1;

case 2:

if(current\_tetromino\_X>=16){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if(current\_tetromino\_Y<0){

cout << "Невохможно повернуться"<<endl;

return 0;

}

if((current\_tetromino\_Y>12) && (turn\_index>=2)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

case 3:

if(current\_tetromino\_X>=16){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_Y<0) && (turn\_index==0)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_Y>12) && (turn\_index>=2)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

case 4:

if(current\_tetromino\_X>=16){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_Y<0)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_Y>12)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

case 5:

if((current\_tetromino\_Y>12)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

case 6:

if((current\_tetromino\_Y>12)){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

}

return 0;

case LEFT:

if((current\_tetromino\_Y==0) && ((current\_tetromino[0][0]==1) || (current\_tetromino[1][0]==1) || (current\_tetromino[2][0]==1) || (current\_tetromino[3][0]==1))){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if(current\_tetromino\_Y ==-1){

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

case RIGHT:

if((current\_tetromino\_Y==11) && (current\_tetromino[1][3]==1)) {

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_Y==12) && ((current\_tetromino[0][2]==1) || (current\_tetromino[1][2]==1) || (current\_tetromino[2][2]==1) || (current\_tetromino[3][2]==1))) {

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

if((current\_tetromino\_Y==13) && ((current\_tetromino[0][1]==1) || (current\_tetromino[1][1]==1) || (current\_tetromino[2][1]==1) || (current\_tetromino[3][1]==1))) {

cout << "Фигура упёрлась в край"<<endl;

return 0;

}

return 1;

}

return 0;

}

int tetromino\_bump(int direction){

int d = 1;

switch(direction){

case DOWN:

for (int i = 0;i<4;i++){

for(int j = 0;j<4;j++){

if(current\_tetromino[i][j]==1){

if(tetris\_field[current\_tetromino\_X+1+i][current\_tetromino\_Y+j]!=0){

return 0;

}

}

}

}

return 1;

case UP:

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

temp\_tetromino[i][j] = current\_tetromino[i][j];

}

}

temp\_turn\_index = turn\_index;

turn\_tetromino();

for (int i = 0;i<4;i++){

for(int j = 0;j<4;j++){

if(current\_tetromino[i][j]==1){

if(tetris\_field[current\_tetromino\_X+i][current\_tetromino\_Y+j]!=0){

d = 0;

break;

}

}

}

if(!d)

break;

}

for (int v =0;v<4;v++){

for (int g=0; g<4;g++){

current\_tetromino[v][g] = temp\_tetromino[v][g];

}

}

turn\_index = temp\_turn\_index;

return d;

case LEFT:

for (int i = 0;i<4;i++){

for(int j = 0;j<4;j++){

if(current\_tetromino[i][j]==1){

if(tetris\_field[current\_tetromino\_X+i][current\_tetromino\_Y-1+j]!=0){

return 0;

}

}

}

}

return 1;

case RIGHT:

for (int i = 0;i<4;i++){

for(int j = 0;j<4;j++){

if(current\_tetromino[i][j]==1){

if(tetris\_field[current\_tetromino\_X+i][current\_tetromino\_Y+1+j]!=0){

return 0;

}

}

}

}

return 1;

}

return 0;

}

int pin\_tetromino(){

score+=10;

if((current\_tetromino\_X==X\_tetromino\_start) && (current\_tetromino\_Y==Y\_tetromino\_start)){

return 1;

}

for (int i = 0;i<4;i++){

for(int j = 0;j<4;j++){

if(current\_tetromino[i][j]==1){

tetris\_field[current\_tetromino\_X+i][current\_tetromino\_Y+j] = current\_color\_index;

}

}

}

DeleteLines();

take\_new\_tetromino();

return 0;

}

void take\_new\_tetromino(){

for (int i =0;i<4;i++){

for (int j=0; j<4;j++){

current\_tetromino[i][j] = next\_tetromino[i][j];

}

}

current\_tetromino\_index = next\_tetromino\_index;

current\_color\_index = next\_color\_index;

generate\_tetromino(next\_tetromino, &next\_color\_index, &next\_tetromino\_index);

turn\_index = 0;

current\_tetromino\_X = X\_tetromino\_start;

current\_tetromino\_Y = Y\_tetromino\_start;

return;

}

int main(){

setlocale(LC\_ALL, "RUS");

srand(time(0));

initwindow(680,550,"Тетрис");//открытие окна

if(tetromino\_blocks\_init() == 0){

return 0;

}

draw\_field();

field\_init();

generate\_tetromino(current\_tetromino, &current\_color\_index, &current\_tetromino\_index);

generate\_tetromino(next\_tetromino, &next\_color\_index, &next\_tetromino\_index);

for (int i=0;i<4;i++){

for (int j=0; j<4;j++){

if(next\_tetromino[i][j]!=0){

draw\_tetromino\_block(YINTERFACE+320+i\*BLOCKSIZE, XINTERFACE+72+j\*BLOCKSIZE, next\_color\_index);

}

}

}

draw\_interface();

enter\_screen();

int i=0;

char KEY;

int end = 0;

while(end==0){

if(move\_time\_passed >= move\_time\_limit){

score++;

char a[100] = "Счёт: ";

char b[90];

sprintf(b, "%d", score);

strcat(a,b);

outtextxy(XSTARTCORD,YSTARTCORD+470, a);

int next = move\_tetromino(DOWN);

if(next){

if(pin\_tetromino()==1){

end = 1;

break;

}

Refresh\_field();

}

move\_time\_passed=0;

}

if(move\_time\_passed == 0){

draw\_tetromino(XTETROMINOSTARTCORD+current\_tetromino\_X\*BLOCKSIZE,YTETROMINOSTARTCORD+current\_tetromino\_Y\*BLOCKSIZE, current\_color\_index);

}

if(kbhit()) {

KEY = getch();

switch(KEY){

case ESC:

return 0;

case ENTER:

outtextxy(XSTARTCORD+120,YINTERFACE+410, "Пауза");

outtextxy(XSTARTCORD+15,YINTERFACE+430, "Нажмите ENTER чтобы продолжить");

while(1){

if(kbhit()){

KEY = getch();

if(KEY == ENTER){

outtextxy(XSTARTCORD+120,YSTARTCORD+410, " ");

outtextxy(XSTARTCORD+15,YSTARTCORD+430, "Нажмите ENTER чтобы остановить");

break;

}

if(KEY == ESC){

return 0;

}

}

}

case DOWN:

move\_time\_passed +=100;

break;

case UP:

move\_tetromino(UP);

break;

case LEFT:

move\_tetromino(LEFT);

break;

case RIGHT:

move\_tetromino(RIGHT);

break;

}

}

delay (5);

move\_time\_passed++;

}

if(end){

setfillstyle ( 1, BLACK );

bar(XSTARTCORD+20, YSTARTCORD+150, XSTARTCORD+280, YSTARTCORD+250);

moveto(XSTARTCORD+20, YSTARTCORD+150);

setcolor(LIGHTRED);

lineto(XSTARTCORD+280,YSTARTCORD+150);

lineto(XSTARTCORD+280,YSTARTCORD+250);

lineto(XSTARTCORD+20,YSTARTCORD+250);

lineto(XSTARTCORD+20,YSTARTCORD+150);

setcolor(WHITE);

settextstyle(10,0,2);

outtextxy(XSTARTCORD+40,YSTARTCORD+180, "ВЫ ПРОИГРАЛИ!");

settextstyle(0,0,1);

outtextxy(XSTARTCORD+30,YSTARTCORD+210, "Нажмите ESC чтобы закончить");

}

while(1){

if(kbhit()){

KEY = getch();

if(KEY == ESC){

break;

}

}

}

free(BlackBlockBuf);

free(GreenBlockBuf);

free(CyanBlockBuf);

free(RedBlockBuf);

free(PurpleBlockBuf);

free(YellowBlockBuf);

closegraph();

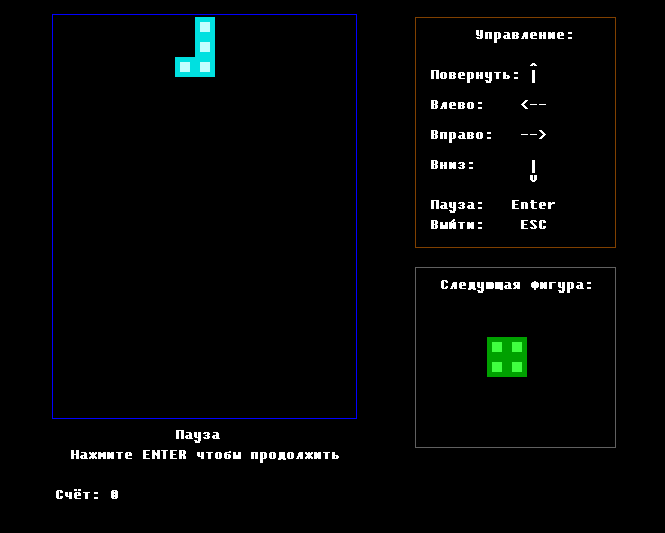
return 0;

}**4. Результат работы программы**

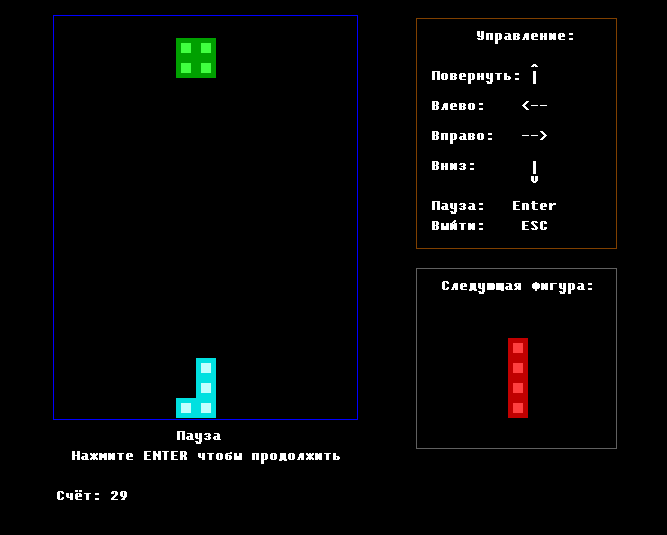
1.начало



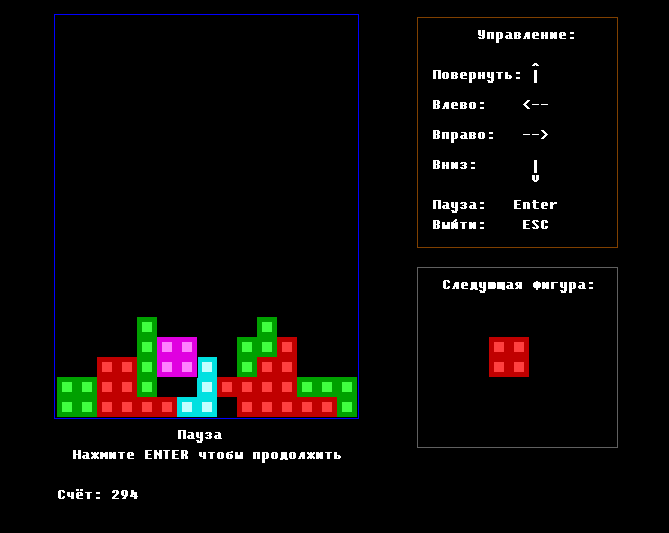
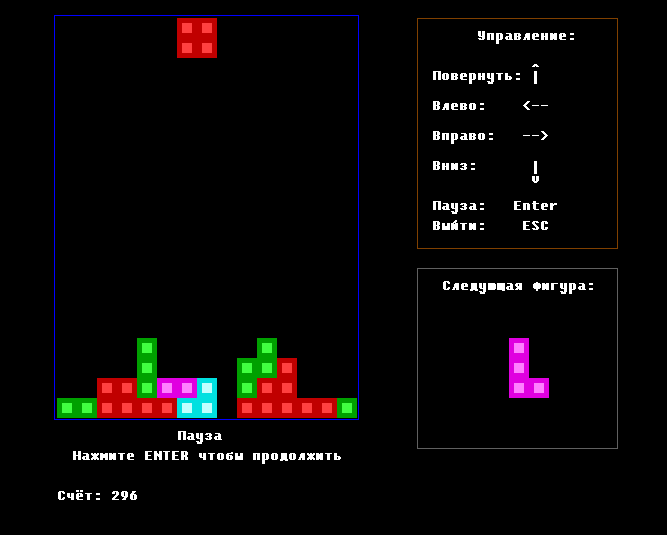
2. После нажатия Enter



3.Когда фигура дошла до границы



4.Исчезновение линии

5.Конец игры

