#include <iostream>

#include <windows.h>

#include <conio.h>

using namespace std;

#pragma comment(lib, "winmm.lib")

#pragma warning (disable:4996)

enum MazeObjects { HALL, WALL, COIN, ENEMY, BORDER, GRAVE, HEALTHBOX }; // MazeObjects - пользовательский (кастомизированный) тип данных

enum Colors { DARKGREEN = 2, YELLOW = 14, RED = 12, BLUE = 9, WHITE = 15, DARKYELLOW = 6, DARKRED = 4, BLACK = 0, MAGENTA = 5, LIGHTMAGENTA = 13, BROWN = 6 };

enum KeyCodes { ENTER = 13, ESCAPE = 27, SPACE = 32, LEFT = 75, RIGHT = 77, UP = 72, DOWN = 80 };

void ShowCoins(int f\_w, int fcoin);

void ShowHealth(int f\_w, int fhealth);

void ShowSteps(int f\_w, int fsteps);

void ShowEnergy(int f\_w, int fenergy);

void StartVisualSettings();

void FillArray(int arr[][40], int fh, int fw);

void ShowMaze(int ar[][40], int fh, int fw);

int HeroPosition(int x, int y);

int main();

DWORD WINAPI LowEnergy(void\* param) // звук мало энергии

{

PlaySoundA("sounds\\lowenergy.wav", NULL, SND\_NODEFAULT | SND\_LOOP);

return 0;

}

DWORD WINAPI Win(void\* param) // звук победы

{

PlaySoundA("sounds\\win.wav", NULL, SND\_NODEFAULT | SND\_LOOP);

return 0;

}

DWORD WINAPI GameOver(void\* param) // звук поражения

{

PlaySoundA("sounds\\gameover.wav", NULL, SND\_NODEFAULT | SND\_LOOP);

return 0;

}

DWORD WINAPI Coin(void\* param) // звук монетки

{

PlaySoundA("sounds\\coin.wav", NULL, SND\_NODEFAULT | SND\_LOOP);

return 0;

}

DWORD WINAPI Kill(void\* param) // звук выстрела

{

PlaySoundA("sounds\\kill.wav", NULL, SND\_NODEFAULT | SND\_LOOP);

return 0;

}

DWORD WINAPI Grave(void\* param) // звук могилка

{

PlaySoundA("sounds\\grave.wav", NULL, SND\_NODEFAULT | SND\_LOOP);

return 0;

}

DWORD WINAPI Medkit(void\* param) // звук аптечка

{

PlaySoundA("sounds\\medkit.wav", NULL, SND\_NODEFAULT | SND\_LOOP);

return 0;

}

int main()

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

int health = 100; // количество очков здоровья главного героя

int coins = 0; // счётчик собранных монет

int steps = 0; // счётчик шагов ГГ

int energy = 120; // счётчик энергии

int answer;

const int WIDTH = 40; // ширина лабиринта (ctrl h замена имени)

const int HEIGHT = 10; // высота лабиринта

int maze[HEIGHT][WIDTH]{};

COORD position; // координаты нашего персонажа

position.X = 0;

position.Y = 2;

StartVisualSettings();

FillArray(maze, HEIGHT, WIDTH);

ShowMaze(maze, HEIGHT, WIDTH);

HeroPosition(position.X, position.Y);

ShowCoins(WIDTH, coins);

ShowHealth(WIDTH, health);

ShowSteps(WIDTH, steps);

ShowEnergy(WIDTH, energy);

while (true)

{

if (\_kbhit()) // keyboard hit если было нажатие на клавиши пользователем

{

int code = \_getch(); // get character, получение кода нажатой клавиши

if (code == 224) { // если это стрелка

code = \_getch(); // получить конкретный код стрелки

}

// стирание ГГ в старой позиции

SetConsoleCursorPosition(h, position);

if (maze[position.Y][position.X] == MazeObjects::GRAVE)

{

SetConsoleTextAttribute(h, Colors::MAGENTA);

cout << "X";

}

else

{

SetConsoleTextAttribute(h, Colors::BLACK);

cout << " ";

}

if (code == KeyCodes::ENTER)

{

if (maze[position.Y][position.X + 1] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y][position.X + 1] = MazeObjects::GRAVE;

COORD effects; // для grave

effects.X = position.X + 1;

effects.Y = position.Y;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

else if (maze[position.Y][position.X - 1] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y][position.X - 1] = MazeObjects::GRAVE;

COORD effects;

effects.X = position.X - 1;

effects.Y = position.Y;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

else if (maze[position.Y + 1][position.X] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y + 1][position.X] = MazeObjects::GRAVE;

COORD effects;

effects.X = position.X;

effects.Y = position.Y + 1;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

else if (maze[position.Y - 1][position.X] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y - 1][position.X] = MazeObjects::GRAVE;

COORD effects;

effects.X = position.X;

effects.Y = position.Y - 1;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

////////////////////////////////////////////////////////////////

else if (maze[position.Y - 1][position.X + 1] == MazeObjects::ENEMY) ////// накрест

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y - 1][position.X + 1] = MazeObjects::GRAVE;

COORD effects;

effects.X = position.X + 1;

effects.Y = position.Y - 1;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

else if (maze[position.Y - 1][position.X - 1] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y - 1][position.X - 1] = MazeObjects::GRAVE;

COORD effects;

effects.X = position.X - 1;

effects.Y = position.Y - 1;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

else if (maze[position.Y + 1][position.X + 1] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y + 1][position.X + 1] = MazeObjects::GRAVE;

COORD effects;

effects.X = position.X + 1;

effects.Y = position.Y + 1;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

else if (maze[position.Y + 1][position.X - 1] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

energy -= 10;

ShowEnergy(WIDTH, energy);

maze[position.Y + 1][position.X - 1] = MazeObjects::GRAVE;

COORD effects;

effects.X = position.X - 1;

effects.Y = position.Y + 1;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::LIGHTMAGENTA);

cout << "X";

}

}

else if (code == KeyCodes::ESCAPE)

{

cout << "ESCAPE!\n";

system("cls");

exit(0);

}

else if (code == KeyCodes::SPACE && energy >= 50)

{

energy -= 50;

ShowEnergy(WIDTH, energy);

int radius = 3;

for (int y = position.Y - radius; y <= position.Y + radius; y++)

{

for (int x = position.X - radius; x <= position.X + radius; x++)

{

if (x >= 0 && x <= WIDTH - 1 &&

y >= 0 && y <= HEIGHT - 1 &&

maze[y][x] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

CreateThread(0, 0, Grave, 0, 0, 0);

maze[y][x] = MazeObjects::GRAVE;

COORD effects;

effects.X = x;

effects.Y = y;

SetConsoleCursorPosition(h, effects);

SetConsoleTextAttribute(h, Colors::MAGENTA);

cout << "X";

}

}

}

}

else if (code == KeyCodes::LEFT

&& maze[position.Y][position.X - 1] != MazeObjects::WALL

&& maze[position.Y][position.X - 1] != MazeObjects::BORDER)

{

position.X--;

}

else if (code == KeyCodes::RIGHT // если я собрался пойти направо

&& maze[position.Y][position.X + 1] != MazeObjects::WALL

&& maze[position.Y][position.X + 1] != MazeObjects::BORDER)

// и при этом в лабиринте на той же строке (где смайлик) и

// немного (на одну ячейку) правее на 1 столбец от смайлика

{

position.X++;

}

else if (code == KeyCodes::UP

&& maze[position.Y - 1][position.X] != MazeObjects::WALL

&& maze[position.Y - 1][position.X] != MazeObjects::BORDER)

{

position.Y--;

}

else if (code == KeyCodes::DOWN

&& maze[position.Y + 1][position.X] != MazeObjects::WALL

&& maze[position.Y + 1][position.X] != MazeObjects::BORDER)

{

position.Y++;

}

// показ ГГ в новой позиции

SetConsoleCursorPosition(h, position);

SetConsoleTextAttribute(h, Colors::BLUE);

cout << (char)2;

// пересечение с элементами массива

if (position.Y == HEIGHT - 3 &&

position.X == WIDTH - 1)

{

CreateThread(0, 0, Win, 0, 0, 0);

answer = MessageBoxA(0, "Найден выход из лабиринта!\n\nХотите сыграть еще?", "ПОБЕДА!", MB\_YESNO);

system("cls");

if (answer == IDYES)

{

coins = 0;

health = 100;

steps = 0;

energy = 120;

main();

}

else exit(0);

}

// пересечение с монетками

// если в лабиринте по позиции смайлика (под ним) находится монетка

if (maze[position.Y][position.X] == MazeObjects::COIN)

{

CreateThread(0, 0, Coin, 0, 0, 0);

coins++; // на одну монетку собрали больше

ShowCoins(WIDTH, coins);

maze[position.Y][position.X] = MazeObjects::HALL; // убираем монетку из лабиринта

energy += 10;

ShowEnergy(WIDTH, energy);

// перебор монеток //

int cur\_coin = 0;

for (int y = 0; y < HEIGHT; y++) // перебор строк

{

for (int x = 0; x < WIDTH; x++) // перебор столбцов

{

if (maze[y][x] == MazeObjects::COIN) cur\_coin++;

}

}

if (cur\_coin == 0)

{

CreateThread(0, 0, Win, 0, 0, 0);

answer = MessageBoxA(0, "Все монеты собраны!\n\nХотите сыграть еще?", "ПОБЕДА!", MB\_YESNO);

system("cls");

if (answer == IDYES)

{

coins = 0;

health = 100;

steps = 0;

energy = 120;

main();

}

else exit(0);

}

}

// пересечение с врагами

// если в лабиринте по позиции смайлика (под ним) находится враг

if (maze[position.Y][position.X] == MazeObjects::ENEMY)

{

CreateThread(0, 0, Kill, 0, 0, 0);

health -= 20;

ShowHealth(WIDTH, health);

maze[position.Y][position.X] = MazeObjects::GRAVE; // убираем монетку из лабиринта

CreateThread(0, 0, Grave, 0, 0, 0);

if (health <= 0)

{

CreateThread(0, 0, GameOver, 0, 0, 0);

answer = MessageBoxA(0, "Здоровье закончилось!\n\nХотите начать сначала?", "ПОРАЖЕНИЕ!", MB\_YESNO);

system("cls");

if (answer == IDYES)

{

coins = 0;

health = 100;

steps = 0;

energy = 120;

main();

}

else exit(0);

}

}

// пересечение с аптечкой //

// если в лабиринте по позиции смайлика (под ним) находится аптечка

if (maze[position.Y][position.X] == MazeObjects::HEALTHBOX)

{

CreateThread(0, 0, Medkit, 0, 0, 0);

health = 100; // добавить сразу обновление инфобокс

ShowHealth(WIDTH, health);

maze[position.Y][position.X] = MazeObjects::HALL; // убираем аптечку из лабиринта

}

if (code == LEFT || code == RIGHT || code == UP || code == DOWN) // подсчёт шагов ГГ и энергии

{

steps++;

ShowSteps(WIDTH, steps);

energy--;

ShowEnergy(WIDTH, energy);

if (energy <= 0)

{

CreateThread(0, 0, GameOver, 0, 0, 0);

answer = MessageBoxA(0, "Энергия закончилась!\n\nХотите начать сначала?", "ПОРАЖЕНИЕ!", MB\_YESNO);

system("cls");

if (answer == IDYES)

{

coins = 0;

health = 100;

steps = 0;

energy = 120;

main();

}

else exit(0);

}

}

}

else // нажатия не было, двигаем врагов

{

Sleep(15);

// движение врагов

COORD enemy\_positions[100]; // массив который хранит координаты врагов

int enemy\_count = 0; // реальное количество врагов на данный момент после шага ГГ

// перебор всех ячеек которые есть в лабиринте

for (int y = 0; y < HEIGHT; y++) // перебор строк

{

for (int x = 0; x < WIDTH; x++) // перебор столбцов

{

if (maze[y][x] == MazeObjects::ENEMY) // если очередная ячейка - это враг

{

enemy\_positions[enemy\_count].X = x;

enemy\_positions[enemy\_count].Y = y;

enemy\_count++;

}

}

}

if (enemy\_count == 0)

{

CreateThread(0, 0, Win, 0, 0, 0);

answer = MessageBoxA(0, "Все враги уничтожены!\n\nХотите сыграть еще?", "ПОБЕДА!", MB\_YESNO);

system("cls");

if (answer == IDYES)

{

coins = 0;

health = 100;

steps = 0;

energy = 120;

main();

}

else exit(0);

}

// перебор всех врагов

for (int i = 0; i < enemy\_count; i++)

{

int r = rand() % 100; // 0-left 1-right 2-up 3-down

if (r == 0 &&

maze[enemy\_positions[i].Y][enemy\_positions[i].X - 1] != MazeObjects::WALL &&

maze[enemy\_positions[i].Y][enemy\_positions[i].X - 1] != MazeObjects::BORDER &&

maze[enemy\_positions[i].Y][enemy\_positions[i].X - 1] != MazeObjects::ENEMY &&

maze[enemy\_positions[i].Y][enemy\_positions[i].X - 1] != MazeObjects::HEALTHBOX) // left

{

// стирание врага в старой позиции

COORD temp = enemy\_positions[i];

temp.X = enemy\_positions[i].X;

temp.Y = enemy\_positions[i].Y;

SetConsoleCursorPosition(h, temp);

cout << " ";

maze[enemy\_positions[i].Y][enemy\_positions[i].X] = MazeObjects::HALL;

// перемещаем врага в новую позицию

temp.X = enemy\_positions[i].X - 1;

temp.Y = enemy\_positions[i].Y;

SetConsoleCursorPosition(h, temp);

SetConsoleTextAttribute(h, Colors::RED);

cout << (char)1;

maze[enemy\_positions[i].Y][enemy\_positions[i].X - 1] = MazeObjects::ENEMY;

}

else if (r == 1 && // right

maze[enemy\_positions[i].Y][enemy\_positions[i].X + 1] != MazeObjects::WALL &&

maze[enemy\_positions[i].Y][enemy\_positions[i].X + 1] != MazeObjects::BORDER &&

maze[enemy\_positions[i].Y][enemy\_positions[i].X + 1] != MazeObjects::ENEMY &&

maze[enemy\_positions[i].Y][enemy\_positions[i].X + 1] != MazeObjects::HEALTHBOX)

{

// стирание врага в старой позиции

COORD temp = enemy\_positions[i];

temp.X = enemy\_positions[i].X;

temp.Y = enemy\_positions[i].Y;

SetConsoleCursorPosition(h, temp);

cout << " ";

maze[enemy\_positions[i].Y][enemy\_positions[i].X] = MazeObjects::HALL;

// перемещаем врага в новую позицию

temp.X = enemy\_positions[i].X + 1;

temp.Y = enemy\_positions[i].Y;

SetConsoleCursorPosition(h, temp);

SetConsoleTextAttribute(h, Colors::RED);

cout << (char)1;

maze[enemy\_positions[i].Y][enemy\_positions[i].X + 1] = MazeObjects::ENEMY;

}

else if (r == 2 && // up

maze[enemy\_positions[i].Y - 1][enemy\_positions[i].X] != MazeObjects::WALL &&

maze[enemy\_positions[i].Y - 1][enemy\_positions[i].X] != MazeObjects::BORDER &&

maze[enemy\_positions[i].Y - 1][enemy\_positions[i].X] != MazeObjects::ENEMY &&

maze[enemy\_positions[i].Y - 1][enemy\_positions[i].X] != MazeObjects::HEALTHBOX)

{

// стирание врага в старой позиции

COORD temp = enemy\_positions[i];

temp.X = enemy\_positions[i].X;

temp.Y = enemy\_positions[i].Y;

SetConsoleCursorPosition(h, temp);

cout << " ";

maze[enemy\_positions[i].Y][enemy\_positions[i].X] = MazeObjects::HALL;

// перемещаем врага в новую позицию

temp.X = enemy\_positions[i].X;

temp.Y = enemy\_positions[i].Y - 1;

SetConsoleCursorPosition(h, temp);

SetConsoleTextAttribute(h, Colors::RED);

cout << (char)1;

maze[enemy\_positions[i].Y - 1][enemy\_positions[i].X] = MazeObjects::ENEMY;

}

else if (r == 3 && // down

maze[enemy\_positions[i].Y + 1][enemy\_positions[i].X] != MazeObjects::WALL &&

maze[enemy\_positions[i].Y + 1][enemy\_positions[i].X] != MazeObjects::BORDER &&

maze[enemy\_positions[i].Y + 1][enemy\_positions[i].X] != MazeObjects::ENEMY &&

maze[enemy\_positions[i].Y + 1][enemy\_positions[i].X] != MazeObjects::HEALTHBOX)

{

// стирание врага в старой позиции

COORD temp = enemy\_positions[i];

temp.X = enemy\_positions[i].X;

temp.Y = enemy\_positions[i].Y;

SetConsoleCursorPosition(h, temp);

cout << " ";

maze[enemy\_positions[i].Y][enemy\_positions[i].X] = MazeObjects::HALL;

// перемещаем врага в новую позицию

temp.X = enemy\_positions[i].X;

temp.Y = enemy\_positions[i].Y + 1;

SetConsoleCursorPosition(h, temp);

SetConsoleTextAttribute(h, Colors::RED);

cout << (char)1;

maze[enemy\_positions[i].Y + 1][enemy\_positions[i].X] = MazeObjects::ENEMY;

}

}

}

}

}

void ShowCoins(int f\_w, int fcoin) // F12 перемещение по функциям в мейне

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD infobox; // инфоблок с показателями

infobox.X = f\_w + 1;

infobox.Y = 1;

SetConsoleCursorPosition(h, infobox);

SetConsoleTextAttribute(h, Colors::DARKYELLOW);

cout << "COINS: ";

SetConsoleTextAttribute(h, Colors::YELLOW);

cout << fcoin << " \n";

}

void ShowHealth(int f\_w, int fhealth)

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD infobox; // инфоблок с показателями

infobox.X = f\_w + 1;

infobox.Y = 2;

SetConsoleCursorPosition(h, infobox);

SetConsoleTextAttribute(h, Colors::DARKRED);

cout << "HEALTH: ";

SetConsoleTextAttribute(h, Colors::RED);

cout << fhealth << " \n";

}

void ShowSteps(int f\_w, int fsteps)

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD infobox; // инфоблок с показателями

infobox.X = f\_w + 1;

infobox.Y = 3;

SetConsoleCursorPosition(h, infobox);

SetConsoleTextAttribute(h, Colors::DARKGREEN);

cout << "STEPS: ";

SetConsoleTextAttribute(h, Colors::WHITE);

cout << fsteps << " \n";

}

void ShowEnergy(int f\_w, int fenergy)

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD infobox; // инфоблок с показателями

infobox.X = f\_w + 1;

infobox.Y = 4;

SetConsoleCursorPosition(h, infobox);

SetConsoleTextAttribute(h, Colors::BLUE);

cout << "ENERGY: ";

SetConsoleTextAttribute(h, Colors::RED);

cout << fenergy << " \n";

if (fenergy == 20)

{

CreateThread(0, 0, LowEnergy, 0, 0, 0);

int answer = MessageBoxA(0, "Низкий уровень энергии!", "ВНИМАНИЕ!", 0);

}

}

void StartVisualSettings()

{

srand(time(0));

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_FONT\_INFOEX cfi; // для работы с шрифтами и тд

cfi.cbSize = sizeof(cfi);

cfi.nFont = 0;

cfi.dwFontSize.X = 0;

cfi.dwFontSize.Y = 40; // можно изменить, чтоб изменился размер окна консоли

cfi.FontFamily = FF\_DONTCARE; // F12 покажет другие варианты

cfi.FontWeight = FW\_NORMAL;

wcscpy(cfi.FaceName, L"Consolas");

SetCurrentConsoleFontEx(GetStdHandle(STD\_OUTPUT\_HANDLE), FALSE, &cfi);

// скрытие мигающего курсора

CONSOLE\_CURSOR\_INFO cursor;

cursor.bVisible = false; // спрятать курсор

cursor.dwSize = 1; // 1...100

SetConsoleCursorInfo(h, &cursor);

system("title Maze"); // имя окна консоли

MoveWindow(GetConsoleWindow(), 20, 60, 1366, 768, true);

// 50 - отступ слева от левой границы рабочего стола до левой границы окна консоли (в пикселях!)

// 60 - отступ сверху от верхней границы РС до верхней границы окна консоли

// 1700 - ширина окна консоли в пикселях

// 1000 - высота окна консоли

}

void FillArray(int arr[][40], int fh, int fw) // алгоритм заполнения массива

{

for (int y = 0; y < fh; y++) // перебор строк

{

for (int x = 0; x < fw; x++) // перебор столбцов

{

arr[y][x] = rand() % 4; // 4 типа объектов в игре

if (arr[y][x] == MazeObjects::ENEMY) // если в лабиринте сгенерился враг

{

int probability = rand() % 15; // 0...9, если выпало 0 - враг останется 01234 останется только одна пятая часть врагов

if (probability != 0) // убираем врага

{

arr[y][x] = MazeObjects::HALL; // на место врага ставим коридор

}

}

if (arr[y][x] == MazeObjects::WALL) // если в лабиринте сгенерилась стена

{

int probability = rand() % 6; // 0...4, если выпало 0 - враг останется 01234 останется только одна пятая часть врагов

if (probability != 0) // убираем стену

{

arr[y][x] = MazeObjects::HALL; // на место стены ставим коридор

}

}

if (x == 0 || y == 0 || x == fw - 1 || y == fh - 1) arr[y][x] = MazeObjects::BORDER;

if (x == 0 && y == 2 || x == 1 && y == 2 || x == 2 && y == 2) arr[y][x] = MazeObjects::HALL; // вход

if (x == fw - 1 && y == fh - 3 ||

x == fw - 2 && y == fh - 3 ||

x == fw - 3 && y == fh - 3) arr[y][x] = MazeObjects::HALL; // выход

if (x == 4 && y == 5) arr[y][x] = MazeObjects::HEALTHBOX; // аптечка

}

}

}

void ShowMaze(int ar[][40], int fh, int fw) // показ лабиринта

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int y = 0; y < fh; y++) // перебор строк

{

for (int x = 0; x < fw; x++) // перебор столбцов

{

switch (ar[y][x])

{

case MazeObjects::HALL: // hall - коридор

SetConsoleTextAttribute(h, Colors::BLACK);

cout << " ";

break;

case MazeObjects::WALL: // wall - стена

SetConsoleTextAttribute(h, Colors::DARKGREEN);

cout << (char)178;

break;

case MazeObjects::BORDER: // border - рамка

SetConsoleTextAttribute(h, Colors::WHITE);

cout << (char)178;

break;

case MazeObjects::COIN: // coin - монетка

SetConsoleTextAttribute(h, Colors::YELLOW);

cout << ".";

break;

case MazeObjects::ENEMY: // enemy - враг

SetConsoleTextAttribute(h, Colors::RED);

cout << (char)1;

break;

case MazeObjects::HEALTHBOX:

SetConsoleTextAttribute(h, 15 \* 16 + 12);

cout << "+";

break;

}

}

cout << "\n";

}

}

int HeroPosition(int x, int y) // размещение главного героя (ГГ)

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD position; // координаты нашего персонажа

position.X = x;

position.Y = y;

SetConsoleCursorPosition(h, position);

SetConsoleTextAttribute(h, Colors::BLUE);

cout << (char)2;

return x, y;

}