Основы программирования Лабораторная работа №10

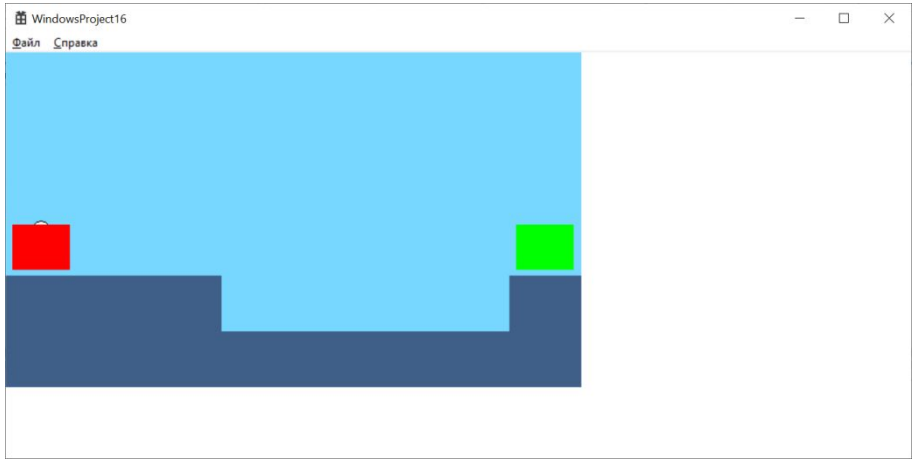
Структуры и игры.

Власенко О.Ф.

Задача сквозная

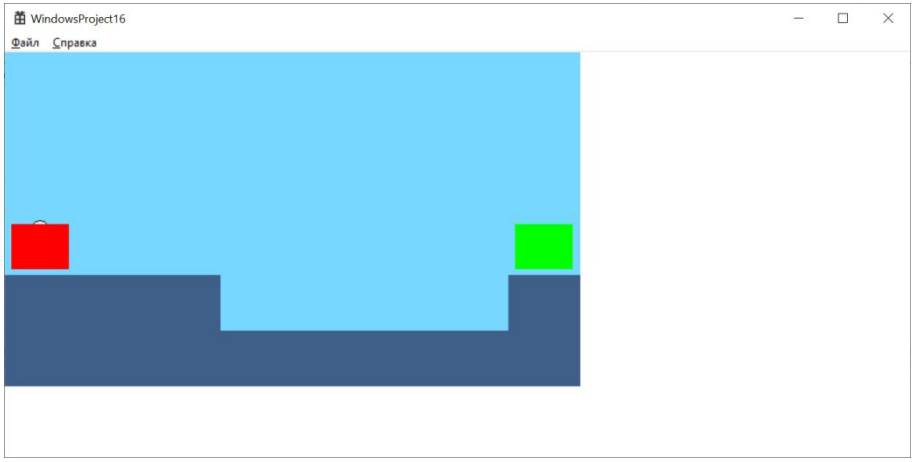
Делаем логику для игры.
В игре есть вертикальная карта – разрез земли (смотри SuperMario и подобные игры).
На этой карте есть элементы двух типов – «воздух» = 0 и «земля» = 1
Есть точка входа в карту – на картинке это красный прямоугольник.
И точка выхода – зеленый прямоугольник.

Ниже приведено отображение возможного состояния игры.



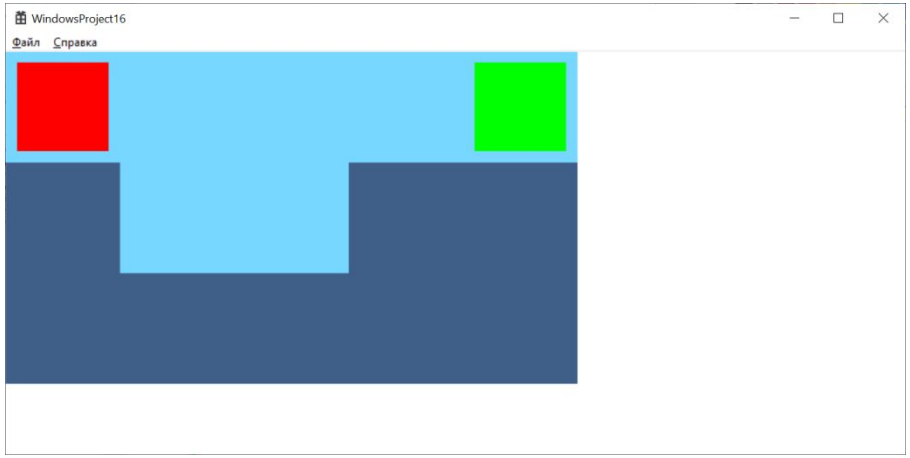
Структура для хранения игры

```
// индексы входа и выхода
struct Position {
	int i, j;
};

// Уровень игры
struct Level {
	int map[10][10]; // карта уровня
// 0 – воздух
// 1 – земля
	int n; // количество строк
	int m; // количество столбцов
	struct Position entry; // вход
	struct Position exit; // выход
};
```

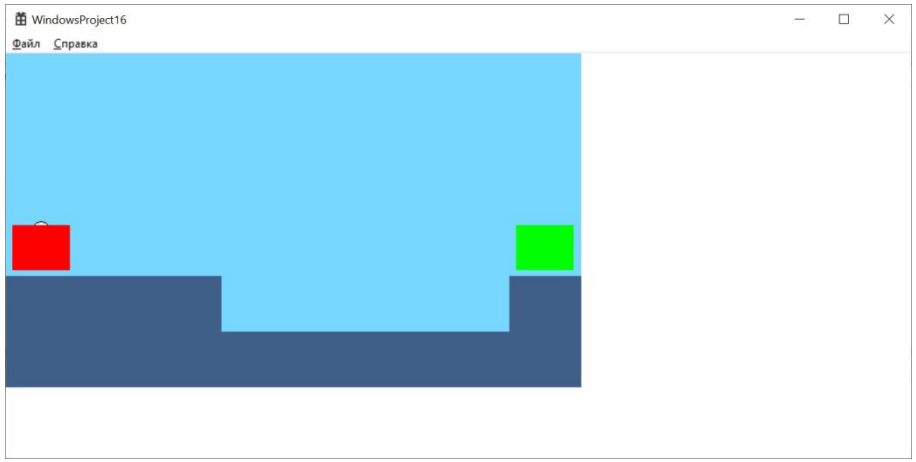


Размеры игрового поля и одного элемента

```
struct Level level =  {
        {
            {0, 0, 0, 0, 0},
            {1, 0, 0, 1, 1},
            {1, 1, 1, 1, 1}
        },
        3,
        5,
        {0, 0},
        {0, 4}
}; // Map
```

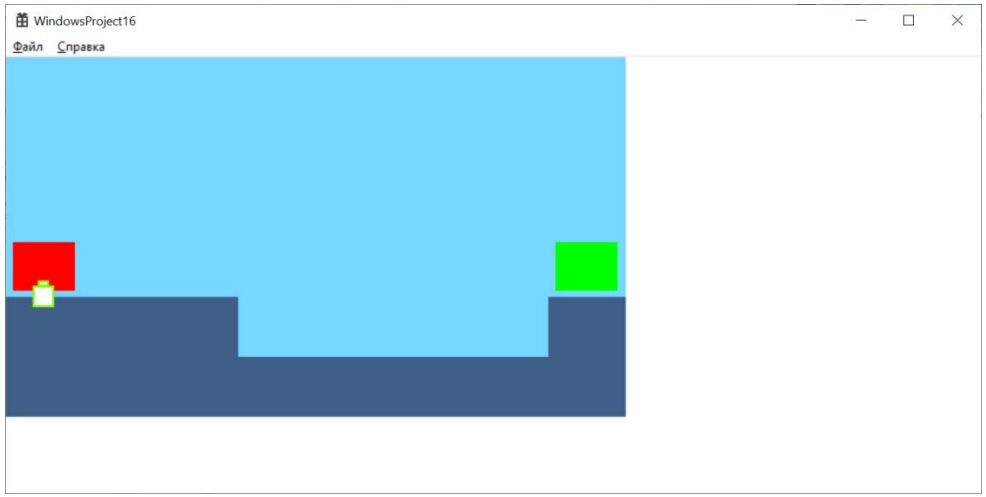



```
int width = 600;
int height = 350;

int sizeX = width / level.m;
int sizeY = height / level.n;
```



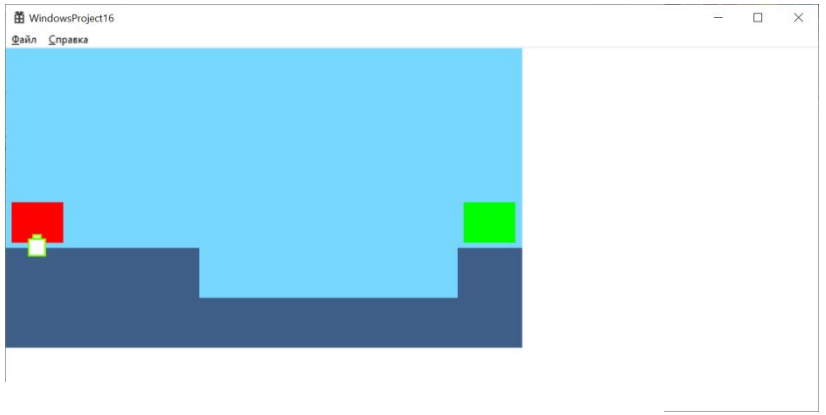
Действующий Херой

```
struct Object {
    POINT pos;
    POINT v ;
};


struct Object hero;
```

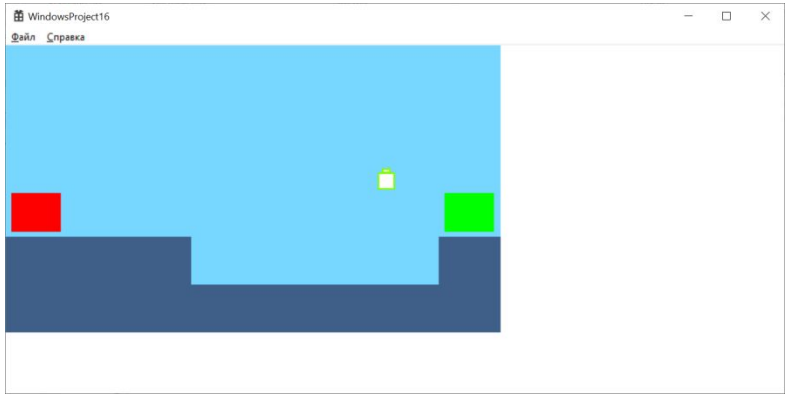


Действующий Херой (2)

Задача Героя – добраться до выхода. С вашей помощью




```
int isFinish() {
    if (hero.pos.y / sizeY == level.exit.i && hero.pos.x / sizeX == level.exit.j) {
        return 1;
    }
    else {
        return 0;
    }
}
```



Действующий Херой (3)

Задача Героя – добраться до выхода. С вашей помощью

```
void MoveHeroLeft() {
    hero.v.x = -5;
}

void MoveHeroRight() {
    hero.v.x = 5;
}

void MoveHeroUp() {
    hero.v.y = -5;
}
```

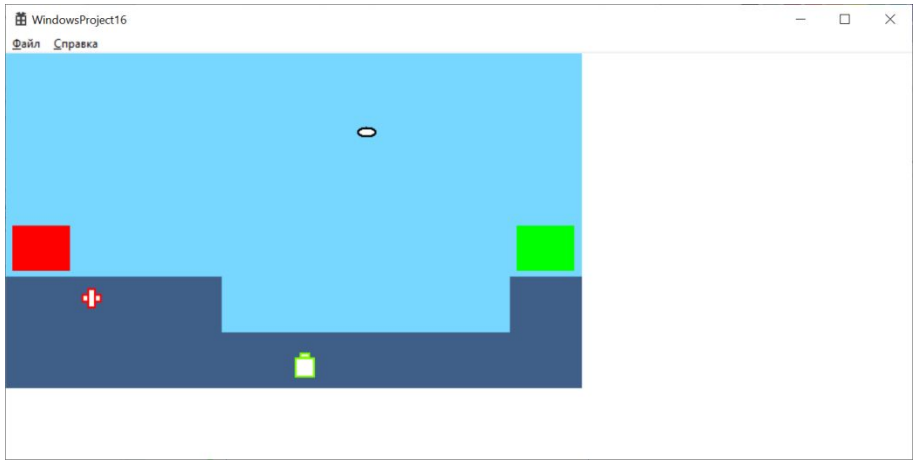
```
void MoveHero() {
    hero.pos.x += hero.v.x;
    hero.pos.y += hero.v.y;

    hero.v.y += 2;

    if (hero.pos.x < 0) {
        hero.pos.x = 0;
    }
    if (hero.pos.x > width) {
        hero.pos.x = width;
    }
    if (hero.pos.y < 0) {
        hero.pos.y = 0;
    }
    if (hero.pos.y > height) {
        hero.pos.y = height;
    }
}
```

Действующий Херой (4)

Задача Героя – добраться до выхода. С вашей помощью

```
case WM_KEYDOWN:
    switch (wParam)
    {
    case 0x4C: // 'L'
        loadLevel(&level);
        sizeX = width / level.m;
        sizeY = height / level.n;
        hero.pos.x = sizeX * level.entry.j + sizeX / 2;
        hero.pos.y = sizeY * level.entry.i + sizeY / 2;

        InvalidateRect(hWnd, NULL, TRUE);
        break;
    case VK_LEFT:
        MoveHeroLeft();
        InvalidateRect(hWnd, NULL, TRUE);
        break;
    case VK_RIGHT:
        MoveHeroRight();
        InvalidateRect(hWnd, NULL, TRUE);
        break;
    case VK_UP:
        MoveHeroUp();
        InvalidateRect(hWnd, NULL, TRUE);
        break;
    }
    break;
```

Летающее нечто



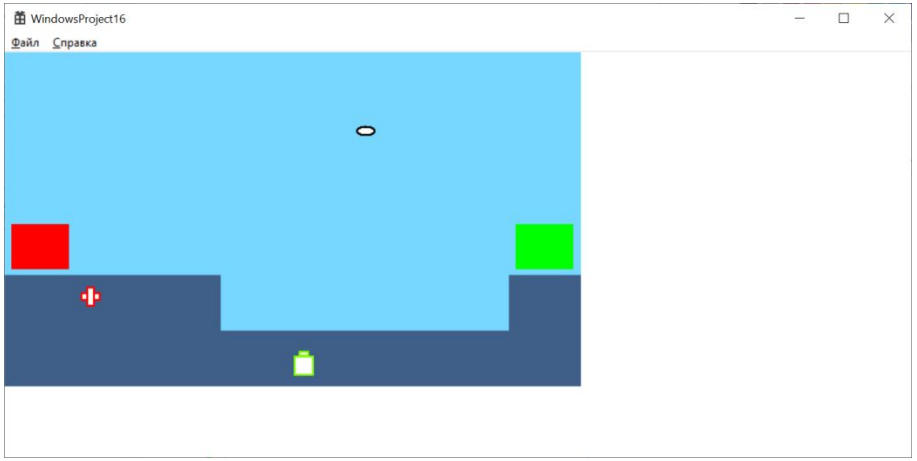

```
struct Object {
    POINT pos;
    POINT v ;
};
```

```
struct Object hero;
struct Object weapon = { 100, 100, 3, -2 };
struct Object health = { 200, 200, 10, -5 };
```

```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код прорисовки, использующий HDC...

        DrawLevel(hdc);
        DrawHero(hdc, &hero);
        DrawWeapon(hdc, &weapon);
        DrawHealth(hdc, &health);

        EndPaint(hWnd, &ps);
    }
    break;
```

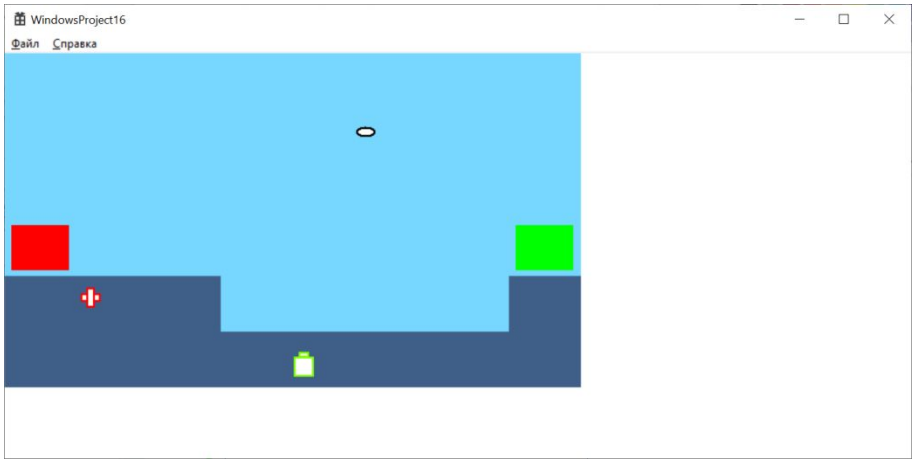
Летающее нечто (2)

```
case WM_CREATE:

    loadLevel(&level);
    sizeX = width / level.m;
    sizeY = height / level.n;
    hero.pos.x = sizeX * level.entry.j + sizeX / 2;
    hero.pos.y = sizeY * level.entry.i + sizeY / 2;

    SetTimer(hWnd, 1, 100, 0);

    break;

case WM_TIMER:
    if (isFinish()) {
        PostQuitMessage(0);
    }
    MoveHero();
    MoveObject(&weapon);
    MoveObject(&health);

    InvalidateRect(hWnd, NULL, TRUE);
    break;
```

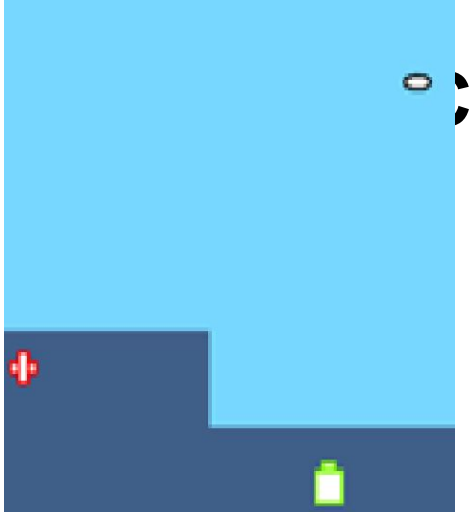


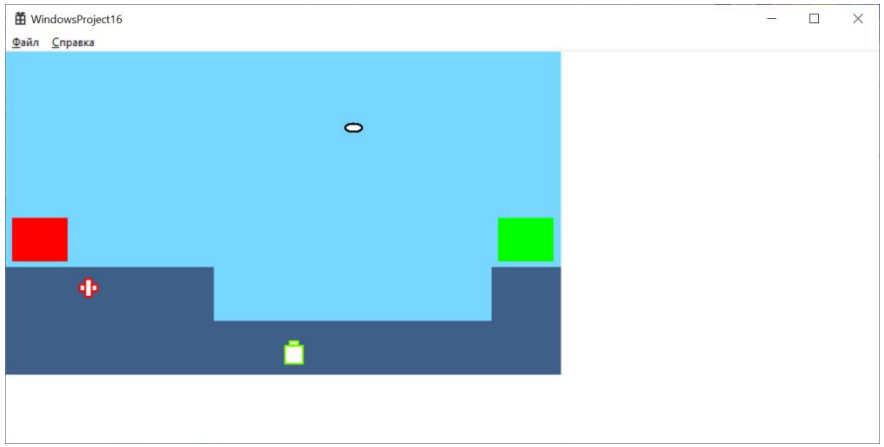
Летающее нечто (3)

```
void MoveObject(struct Object * obj) {
    obj->pos.x += obj->v.x;
    obj->pos.y += obj->v.y;

    if (obj->pos.x < 0) {
        obj->pos.x = 0;
        obj->v.x = -obj->v.x;
    }

    if (obj->pos.x > width) {
        obj->pos.x = width;
        obj->v.x = -obj->v.x;
    }

    if (obj->pos.y < 0) {
        obj->pos.y = 0;
        obj->v.y = -obj->v.y;
    }

    if (obj->pos.y > height) {
        obj->pos.y = height;
        obj->v.y = -obj->v.y;
    }
}
```



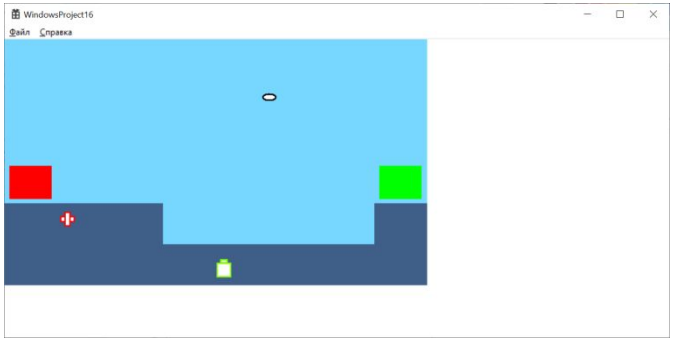
Собственно отрисовка всего





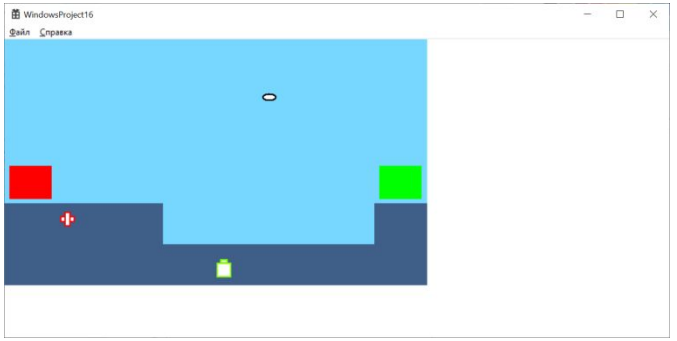

```
HPEN hPenHero = CreatePen(PS_SOLID, 2, RGB(128, 255, 0));
void DrawHero(HDC hdc, struct Object * obj) {
    SelectObject(hdc, hPenHero);

    Rectangle(hdc, obj->pos.x - 10, obj->pos.y - 10, obj->pos.x + 10, obj->pos.y + 10);
    Rectangle(hdc, obj->pos.x - 5, obj->pos.y - 15, obj->pos.x + 5, obj->pos.y - 10);

}

HPEN hPenWeapon = CreatePen(PS_SOLID, 2, RGB(0, 0, 0));
void DrawWeapon(HDC hdc, struct Object * obj) {
    SelectObject(hdc, hPenWeapon);
    Ellipse(hdc, obj->pos.x - 10, obj->pos.y - 5, obj->pos.x + 10, obj->pos.y + 5);
}

HPEN hPenHealth = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
void DrawHealth(HDC hdc, struct Object * obj) {
    SelectObject(hdc, hPenHealth);

    Rectangle(hdc, obj->pos.x - 10, obj->pos.y - 4, obj->pos.x + 10, obj->pos.y + 4);
    Rectangle(hdc, obj->pos.x - 4, obj->pos.y - 10, obj->pos.x + 4, obj->pos.y + 10);
}
```

Собственно отрисовка всего (2)




```
HBRUSH hBrush[2] = {
    CreateSolidBrush(RGB(119, 215, 255)),
    CreateSolidBrush(RGB(63, 95, 137))
};

HBRUSH hBrushEntry = CreateSolidBrush(RGB(255, 0, 0));
HBRUSH hBrushExit = CreateSolidBrush(RGB(0, 255, 0));

void DrawLevel(HDC hdc) {

    int i, j;
    for (i = 0; i < level.n; i++) {
        for (j = 0; j < level.m; j++) {
            RECT rect = {
                j * sizeX,  i * sizeY,
                (j + 1) * sizeX,    (i + 1) * sizeY
            };
            FillRect(hdc, &rect, hBrush[level.map[i][j]]);
        }
    }
```

Собственно отрисовка всего (3)




```
    RECT rectEntry = {
        level.entry.j * sizeX + sizeX * 0.1,    level.entry.i * sizeY + sizeY * 0.1,
        (level.entry.j + 1) * sizeX - sizeX * 0.1,  (level.entry.i + 1) * sizeY - sizeY * 0.1
    };
    FillRect(hdc, &rectEntry, hBrushEntry);


    RECT rectExit = {
        level.exit.j * sizeX + sizeX * 0.1, level.exit.i * sizeY + sizeY * 0.1,
        (level.exit.j + 1) * sizeX - sizeX * 0.1,   (level.exit.i + 1) * sizeY - sizeY * 0.1
    };
    FillRect(hdc, &rectExit, hBrushExit);

}
```

Задача 1. Запустить игру и разобраться в исходниках

Файл с исходными текстами игры предоставлен (на флешке).
Нужно используя его собрать работающую версию.
После получения работающей версии нужно разобраться в исходниках

Задача 2. Добавить возможности в игру

Подключить через нажатие клавиш одну из ранее реализованных возможностей редактирования карты:

- **Добавить элемент земли в заданный столбец**
- **Подсыпать земли в низину**
- **Удалить «землю» из заданной горки**
- **Срыть самую высокую горку**
- **Сохранить состояние игры в файле**

Задача 3. Добавить возможности в игру

Добавить к игре одну или несколько возможностей из следующих:

- **На игровом поле не одно здоровье, а много (массив!)**
- **На игровом поле не одно оружие, а много (массив!)**
- **Попадание оружия в героя делает ему плохо. Через какое то количество шагов он умирает и игра заканчивается***
- **Попадание здоровья в героя делает ему хорошо. ***
- **После прохождения одного уровня, автоматически загружается второй****
- **Герой не проваливается под землю, а ходит по ней*****

Домашнее задание №10

Домашняя работа по лабораторной работе №10 включает в себя

1) Реализация домашней работы №9 в варианте GUI. Нужно визуально отобразить операции над массивами. Разные значения элементов отображаются ра

2) Подготовить отчет (со стандартным содержанием - титульный лист, задание, распечатка, блоксхемы методов (не надо делать, если взялись делать свою игру!))

Домашнее задание №9 - варианты

Вариант 1:
В массиве все элементы, стоящие выше максимального элемента, заменить на максимальный элемент первого столбца.

Вариант 2:
В массиве все элементы, стоящие выше максимального элемента, заменить на минимальный элемент последней строки.

Вариант 3:
В массиве все элементы, стоящие выше и левее минимального элемента, заменить на среднее арифметическое минимального и максимального элементов.

Вариант 4:
В массиве все элементы, стоящие ниже и левее максимального элемента, заменить на среднее арифметическое минимального и максимального элементов последнего столбца.

Вариант 5:
В массиве все элементы, стоящие ниже и левее максимального элемента, заменить на минимальный элемент.

Вариант 6:
В массиве все нечетные элементы, стоящие ниже минимального элемента массива и стоящие слева от максимального элемента массива, заменить на 0.

Вариант 7:
В массиве все четные элементы, стоящие снизу от максимального элемента массива, заменить на максимальный элемент столбца, в котором они расположены.

Вариант 8:
В массиве все нечетные элементы, стоящие сверху от минимального элемента массива, заменить на максимальный элемент строки, в которой они расположены.

Вариант 9:
В массиве все элементы, имеющие четное значение суммы индексов, заменить на минимальный элемент массива.

Вариант 10:
Обнулить элементы в тех столбцах, в которых встречается хотя бы два одинаковых элемента.

Альтернативные варианты – смотри

Домашнее задание №9 – альтернативные варианты

Выберите себе игру, которую вы хотели бы реализовать.
Игра должна быть такой, чтобы наилучшая ее реализация была на основе двухмерного массива.

Реализуйте основной алгоритм из этой игры.

Вдохновение можно получить изучая этот документ:
https://docs.google.com/document/d/1tHW6VXBzvQb8nYH_AVJS3KDsoXh00K7GO2ZF6LZeBS8/edit?usp=sharing