МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «ООП»

Тема: «Создание игрового поля»

Студентка гр. 9383 _____ Лапина А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Научиться создавать игрока и элементов для поля, используя библиотеку SFML, получение знаний об объектно-ориентированном программировании.

Задание.

Создан класс игрока, которым управляет пользователь. Объект класса игрока может перемещаться по полю, а также взаимодействовать с элементами поля. Для элементов поля должен быть создан общий интерфейс и должны быть реализованы 3 разных класса элементов, которые по разному взаимодействуют с игроком. Для взаимодействия игрока с элементом должен использоваться перегруженный оператор *(Например, оператор +)*. Элементы поля могут добавлять очки игроку/замедлять передвижения/и.т.д.

Обязательные требования:

- Реализован класс игрока
- Реализованы три класса элементов поля
- Объект класса игрока появляется на клетке со входом
- Уровень считается пройденным, когда объект класса игрока оказывается на клетке с выходом (и при определенных условиях: например, набрано необходимое кол-во очков)
- Взаимодействие с элементами происходит через общий интерфейс
- Взаимодействие игрока с элементами происходит через перегруженный оператор

Дополнительные требования:

- Для создания элементов используется паттерн **Фабричный метод/Абстрактная фабрика**

- Реализовано динамическое изменение взаимодействия игрока с элементами через паттерн **Стратегия**. Например, при взаимодействии с определенным количеством элементов, игрок не может больше с ними взаимодействовать

Выполнение работы.

1) Создание класса игрока - Player:

создаем поля:

int x_now — координата по x, где находится игрок;

int y_now — координата по y, где находится игрок;

int money - показывает количество монет;

int live — показывает количество жизней;

bool tele = true - переменная, которая принимает значение «false», когда телепорт использован (его можно использовать только 1 раз, после использования он исчезает);

методы:

Player() - конструктор;

int GetLive() - возвращает количество жизней;

int GetMoney() - возвращает количество монет;

int GetX() - возвращает текущую позицию по x;

int GetY() - возвращает текущую позицию по y;

int Put_Money() - добавляет монет;

int Pull_Live() - уменьшает жизни;

void Go(int x_go, int y_go) — метод для передвижения по полю;

void teleport() - функция телепорта;

2) Добавления в класс Game:

В данный класс были добавлены текстуры:

```
texture[0].loadFromFile("player.jpg");//текстура игрока
texture[1].loadFromFile("trava.jpg");//текстура травы
texture[2].loadFromFile("money.jpg");//текстура монеты
```

texture[3].loadFromFile("teleport.jpg");//текстура телепорт

texture[4].loadFromFile("stone.jpg");//текстура камня

В данном классе идею взаимодействие игры с пользователем. Управлять игроком можно с помощью клавиш - «W» - движение вверх, «A» - влево, «S» - вниз, «D» - вправо. Цель игры — добраться до выхода — красной клетки в нижнем правом углу поля, при этом собрав максимальное количество монет и наткнувшись на минимальное количество камней.

Разработанный программный код и см. в приложении А.

Тестирование.

При запуске программы появляется поле (рис. 1). На рис. 2 изображена UML-диаграмма.

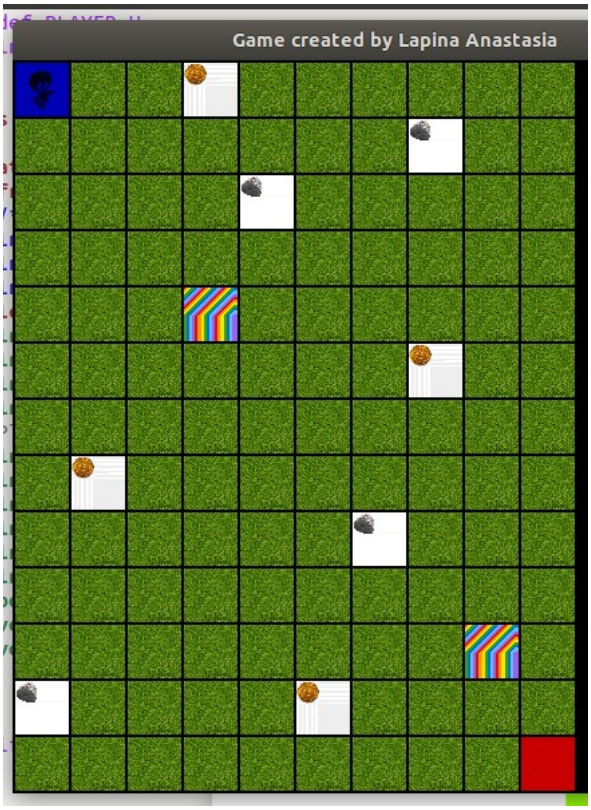

Рис. 1

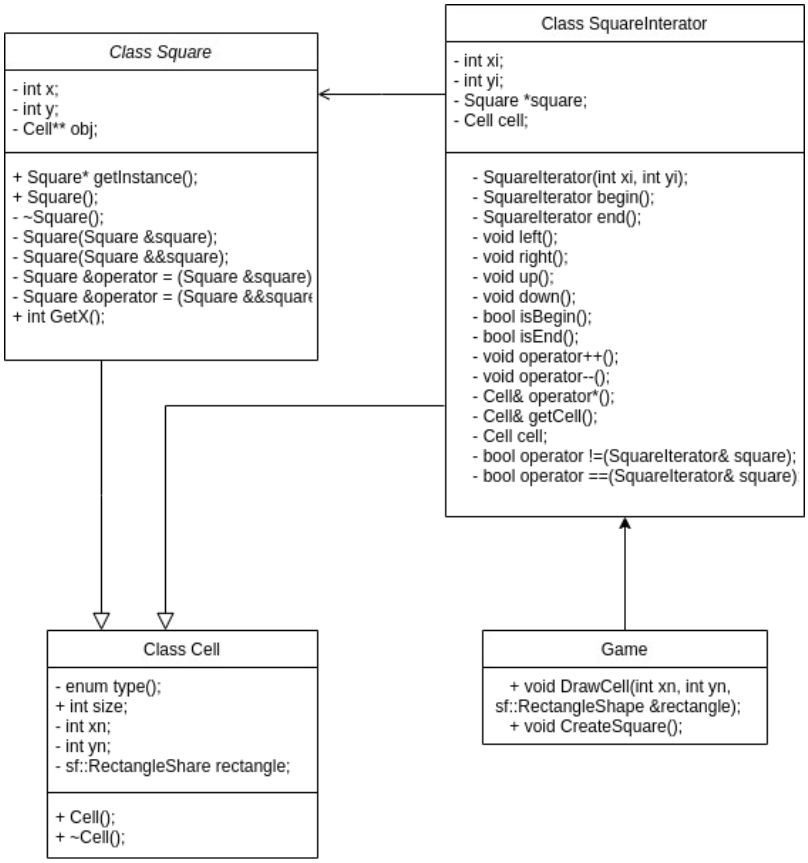

рис. 2

Вывод.

Были получены знания об объектно-ориентированном программировании, были созданы игрок и элементы поля, используя библиотеку SFML.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include "cell.h"
#include "square.h"
#include "squareiterator.h"
#include "game.h"

int main()
{
   Game game;
   game.CreateSquare();
   return 0;
}
```

Название файла: cell.cpp

```
#include "cell.h"

Cell::Cell(){
   this->xn = xn;
   this->yn = yn;
}
```

Название файла: cell.h

```
#ifndef CELL_H
```

```
#define CELL_H

#include <SFML/Graphics.hpp>

class Cell
{
private:
    enum type {o, i, e};
    int xn;
    int yn;
    sf::RectangleShape rectangle;
public:
    int size = 70;
    Cell();
    ~Cell(){};
};

#endif // CELL_H
```

Название файла: square.cpp

```
 #include "square.h"

//конструктор
Square::Square(int x, int y)
{
    {
        obj = new Cell **[x];
        for (int i = 0; i < x; i++){
            obj[i] = new Cell *[y];
```

```
            for (int j = 0; j < y; j++){
                obj[i][j] = nullptr;
            }

        }
    }
}

//конструктор копирования
Square::Square(Square &square): x(square.x), y(square.y), obj(square.obj){
    obj = new Cell **[x];
    for (int i = 0; i < x; i++){
        obj[i] = new Cell *[y];
        for (int j = 0; j < y; j++){
            obj[i][j] = nullptr;
        }
    }

for (int i = 0; i < x; ++i){
    for (int j = 0; j < y; ++j){
        obj[i][j] = square.obj[i][j];
    }
}
}

//конструктор перемещения
Square::Square(Square &&square): x(square.x), y(square.y), obj(square.obj)
{

    obj = new Cell **[x];
```

```
   for (int i = 0; i < x; i++){
       obj[i] = new Cell *[y];
       for (int j = 0; j < y; j++){
          obj[i][j] = nullptr;
       }
    }
       for(int i = 0; i < x; i++){
          swap(obj[i], square.obj[i]);
       }
       swap(obj, square.obj);
}

//деструктор
Square::~Square()
{
   for (int i = 0; i < this-> x; i++) {
       delete [] this->obj[i];
    }
    delete [] this->obj;
}

Square* Square::singleton_= nullptr;

Square* Square::getInstance(int x, int y)
{
    if (!singleton_)
    {
       singleton_ = new Square(x, y);
    }
    return singleton_;
```

```
}

int Square::GetX()
{
   return x;
}

int Square::GetY()
{
   return y;
}
```

Название файла: square.h

```
   #ifndef SQUARE_H
#define SQUARE_H

#include "cell.h"

using namespace std;

class Square
{
private:
   friend class Game;
   friend class SquareIterator;
   int x = 10;
   int y = 13;
   Cell** *obj;
   static Square* singleton_;
```

```
Square(int x, int y); //конструктор
~Square(); //деструктор

Square(Square &square); //конструктор копирования
Square(Square &&square);//конструктор перемещения
//оператор копирования
Square &operator=(Square &square){
   //Исключение самоприсваивания
   if (&square == this)
      return *this;
   //очищаем память (указатели до исключения)
   for (int i = 0; i < y; ++i){
      delete (obj[i]);
   }
   delete (obj);
 //Копирование
 obj = new Cell **[x];
 for (int i = 0; i < x; i++) {
    obj[i] = new Cell *[y];
    for (int j = 0; j < y; j++) {
       obj[i][j] = nullptr;
    }
 }
 for (int i = 0; i < x; ++i) {
    for (int j = 0; j < y; ++j) {
       obj[i][j] = square.obj[i][j];
    }
 }
 return *this;
```

```
}

    //оператор перемещения
    Square &operator=(Square &&square){
    //исключение самоприсваивания
    if (&square == this)
        return *this;
    //очищаем память (указатели до исключения)
    this->x = square.x;
    this->y = square.y;
    this->obj = square.obj;
    return *this;
    }

public:

    static Square* getInstance(int x, int y);
    Square(){};
    int GetX();
    int GetY();
};

#endif // SQUARE_H
```

Название файла: squareiterator.cpp

```
#include "squareiterator.h"

SquareIterator::SquareIterator(int xi, int yi){
    square = Square::getInstance(xi, yi);
```

```
        if (xi >= 0 && square -> GetX() >xi && yi >= 0 && square -> GetY() >
yi){
            this->xi = xi;
            this->yi = yi;
        }
        else
            std::cout << "Invalid coordinates!" << "\n";

    }

    SquareIterator SquareIterator::begin(){
        return SquareIterator(0, 0);
    }

    SquareIterator SquareIterator::end(){
        Square square;
        return SquareIterator(square.x - 1, square.y - 1);
    }

    void SquareIterator::left(){
        if (xi!=0){
            xi = xi - cell.size;
        }
    }

    void SquareIterator::right(){
        if (xi!= square -> GetX() - cell.size){
            xi = xi + cell.size;
        }
    }
```

```
void SquareIterator::up(){
   if (yi!=0){
      yi = yi - cell.size;
   }
}

void SquareIterator::down(){
   if (yi!= square -> GetY() - cell.size){
      yi = yi - cell.size;
   }
}

Cell& SquareIterator::getCell(){
   return cell;
}

bool SquareIterator::isBegin(){
   return (xi == 0 && yi == 0);
}

bool SquareIterator::isEnd(){
      return (xi == square -> GetY() - cell.size && yi == square->GetX() - cell.size);
}

void SquareIterator::operator++(){
   right();
}
```

```
void SquareIterator::operator--(){
   left();
}

Cell& SquareIterator::operator*(){
   return getCell();    //возвращает ссылку на клетку
}

bool SquareIterator::operator !=(SquareIterator &square){
   if (this->xi!=square.xi && this->yi!=square.yi)
      return true;
   else
      return false;
}

bool SquareIterator::operator ==(SquareIterator &square){
   if (this->xi==square.xi && this->yi==square.yi)
      return true;
   else
      return false;
}
```

Название файла: squareiterator.h

```
#ifndef SQUAREITERATOR_H
#define SQUAREITERATOR_H

#include "cell.h"
#include "square.h"
#include "squareiterator.h"
```

```
#include <iostream>

class SquareIterator
{
private:
    int xi;
    int yi;
    Square *square;
    SquareIterator(int xi, int yi);
    SquareIterator begin();
    SquareIterator end();
    void left();
    void right();
    void up();
    void down();
    bool isBegin();
    bool isEnd();
    void operator++();
    void operator--();
    Cell& operator*();
    Cell& getCell();
    Cell cell;
    bool operator !=(SquareIterator& square);
    bool operator ==(SquareIterator& square);

};
#endif // SQUAREITERATOR_H
```

Название файла: game.cpp

```
#include "game.h"

void Game::CreateSquare(){
    sf::RenderWindow window(sf::VideoMode(x*size+22+150, y*size+28), "Game created by Lapina Anastasia");
    texture[0].loadFromFile("player.jpg");//текстура игрока
    texture[1].loadFromFile("trava.jpg");//текстура травы
    texture[2].loadFromFile("money.jpg");//текстура монеты
    texture[3].loadFromFile("teleport.jpg");//текстура телепорт
    texture[4].loadFromFile("stone.jpg");//текстура камня


    rect = new sf::RectangleShape *[x];
    for (int i = 0; i < x; i++) {
        rect[i] = new sf::RectangleShape[y];
        for (int j = 0; j < y; j++) {
            if (i==0 && j==0){
                rect[i][j].setFillColor(sf::Color(0, 0, 200));
            }
            if (i == x-1 && j == y-1){
                rect[i][j].setFillColor(sf::Color(200, 0, 0));
                //rect[i][j].setTexture(&texture[1]);
            }
            if ((i!=0 || j!=0) && (i!=x-1 || j!=y-1 )){
                //rect[i][j].setFillColor(sf::Color(255, 255, 255));
                rect[i][j].setTexture(&texture[1]);
            }
            rect[i][j].setSize(sf::Vector2f(size, size));
            rect[i][j].setPosition(sf::Vector2f(size * i + 2*(i+1), size * j + 2*(j+1)));
```

```
    }
  }
  rect[3][0].setTexture(&texture[2]); //монеты
  rect[1][7].setTexture(&texture[2]); //монеты
  rect[7][5].setTexture(&texture[2]); //монеты
  rect[5][11].setTexture(&texture[2]);//монеты

  rect[3][4].setTexture(&texture[3]); //телепорт
  rect[8][10].setTexture(&texture[3]);//телепорт

  rect[7][1].setTexture(&texture[4]); //камень
  rect[4][2].setTexture(&texture[4]); //камень
  rect[6][8].setTexture(&texture[4]); //камень
  rect[0][11].setTexture(&texture[4]);//камень

  drawPlayer();

  money.setSize(sf::Vector2f(100, 100));
  money.setPosition(sf::Vector2f(600, 600));
  money.setFillColor(sf::Color(20, 100, 10));
 // window.draw(money);

  while (window.isOpen())
  {
     sf::Event event;
     while (window.pollEvent(event))
     {
        if (event.type == sf::Event::Closed)
           window.close();
           if (event.type == sf::Event::KeyPressed) {
```

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::W)){
        clearCell(player.x_now, player.y_now);
        player.Go(-1, 0);
        drawPlayer();
}
if (sf::Keyboard::isKeyPressed(sf::Keyboard::A)){
      clearCell(player.x_now, player.y_now);

        player.Go(0, -1);
        drawPlayer();
}
if (sf::Keyboard::isKeyPressed(sf::Keyboard::S)){
        clearCell(player.x_now, player.y_now);
        player.Go(1, 0);
        drawPlayer();
}
if (sf::Keyboard::isKeyPressed(sf::Keyboard::D)){
      clearCell(player.x_now, player.y_now);
        player.Go(0, 1);
        drawPlayer();
}
   if (player.tele){
      player.teleport();
      if (!player.tele)
         clearCell(4, 3);
}
   drawPlayer();
   if (player.x_now==y-1 && player.y_now==x-1)
      window.close();
   }
```

```
        }


        window.clear();
        window.draw(money);
        //SquareInterator *s;
          for (int i = 0; i < x; i++)
                  for (int j = 0; j < y; j++)
                          window.draw(rect[i][j]);

        window.display();
    }

    for (int i = 0; i < x; i++)
         delete [] rect[i];
    delete [] rect;
    delete square;
}


void Game::drawPlayer(){
        rect[player.y_now][player.x_now].setTexture(&texture[0]);

}

void Game::clearCell(int x_c, int y_c){

         rect[y_c][x_c].setTexture(&texture[1]);

}
```

Название файла: game.h

```
#ifndef GAME_H
#define GAME_H

#include <SFML/Graphics.hpp>

#include "cell.h"
#include "square.h"
#include "squareiterator.h"
#include "player.h"

class Game
{
public:
    int x = 10;
    int y = 13;
    int size = 40;

    Square *square = Square::getInstance(x, y);
    sf::RectangleShape **rect;
    sf::RectangleShape money;
    sf::Texture texture[5];
    Player player;
   // Game()=default;
    void CreateSquare();
    void drawPlayer();
    void clearCell(int x_c, int y_c);
};
```

```
#endif // GAME_H
```

Название файла player.cpp

```
#include "player.h"

Player::Player()
{
    x_now = 0;
    y_now = 0;
    money = 0;
    live = 3;
}

int Player::GetLive(){
    return live;
}

int Player::GetMoney(){
    return money;
}

int Player::Put_Money(){
    money++;
    return money;
}

int Player::Pull_Live(){
    live--;
```

```
    return live;
}

int Player::GetX(){
    return x_now;
}

int Player::GetY(){
    return y_now;
}

void Player::teleport()
{
    if (x_now==4 && y_now==3){
        tele = false;
        x_now = 10;
        y_now = 8;
    }
}

void Player::Go(int x_go, int y_go){
    if (x_now>0&&x_now<12){
        x_now = x_now+x_go;
    }
    if (x_now==0&&x_go>0)
        x_now = x_now+x_go;

    if (x_now==12&&x_go<0)
        x_now = x_now+x_go;
```

```
    if (y_now>0&&y_now<9){
        y_now = y_now+y_go;
    }
    if (y_now==0&&y_go>0)
        y_now = y_now+y_go;
    if (y_now==9&&y_go<0)
        y_now = y_now+y_go;
}
```

Название файла player.h

```
#ifndef PLAYER_H
#define PLAYER_H


class Player
{
private:
    friend class teleport;
    /*int x_now;
    int y_now;
    int money;
    int live;*/
public:
    int x_now;
    int y_now;
    int money;
    int live;
    Player();
    int GetLive();
```

```
    int GetMoney();
    int GetX();
    int GetY();
    int Put_Money(); //добавить монет
    int Pull_Live(); //уменьшить жизни
    bool tele = true;
    void Go(int x_go, int y_go);
    void teleport();

};

#endif // PLAYER_H
```