**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Объекто-ориентированное программирование»**

Тема: Конструкторы и деструкторы

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2381 |  | Двиков Д.В. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

## Цель работы

Изучить работу конструкторов и деструкторов в с++. Реализовать класс поля и конструкторы перемещения и копирования к нему.

## Задание

а) Создать класс клетки игрового поля. Клетка игрового поля может быть проходимой или нет, тем самым определяя возможность игрока встать на эту клетку. Возможность задать проходимость клетки должна быть реализована через конструктор и через метод клетки. В будущем в клетке будет храниться указатель на интерфейс события.

б) Создать класс игрового поля. Игровое поле представляет собой прямоугольник из клеток (двумерный массив). В учебных целях, клетки хранятся как чистый массив на указателях (использовать контейнеры stl запрещено в этой лаб. работе). Размер поля передается в конструктор поля, в котором динамически выделяется память под массив клеток. Также должна быть возможность вызвать конструктор поля без аргументов. Так как происходит выделение память, то необходимо реализовать деструктор в котором будет происходить очистка память. Также для класса поля необходимо реализовать конструкторы копирования и перемещения, а также соответствующие им операторы присваивания.

Также добавить в игровое поле добавьте информацию о входе (где в начале появляется игрок) и выходе (куда игрок должен дойти)

в) В класс управления игрока добавить взаимодействия с полем. При перемещении игрока должна быть проверка на проходимость клетки, если клетка непроходима, то перемещение не должно производиться.

Примечания:

Так как в клетке будет храниться указатель, то при копировании и перемещении должен быть предусмотрен механизм копирования объекта хранящегося по указателю.

Убедитесь, что присутствует проверка контроля размера поля, чтобы его нельзя было сделать слишком маленьким или с отрицательными размерами.

В конструкторе перемещения и соответствующем ему операторе присваивания не должно происходить никакого копирования данных.

Через класс поля должен быть доступ к каждой индивидуальной клетке. Создавать метод, который возвращает указатель на весь массив или указатель на каждую клетку, плохая практика, так как появляется возможность очистки памяти.

**­**

## Выполнение работы

*Класс “Cell” (cell.hpp, cell.cpp)*: Класс клетки. Поле данных: **cellPassible** (логическое значение, указывающее, проходима ли клетка).

Методы:

* + **Cell(bool cellPassible = true):** Конструктор, инициализирует состояние клетки.
  + **bool getCellPassible() const**: Возвращает состояние проходимости клетки.
  + **void setCellPassible(bool cellPassible):** Устанавливает состояние проходимости клетки.

Класс “Gamefield” (gamefield.hpp, gamefield.cpp):

Поля данных:

**gamefield** (двумерный массив клеток, представляющий игровое поле).

**width** и **height** (ширина и высота игрового поля).

**start** и **finish** (пары координат, обозначающие начальную и конечную точки на поле).

Методы:

**Gamefield(int width = D\_WIDTH, int height = D\_HEIGHT):** Конструктор, создает игровое поле с указанными параметрами.

**Gamefield(Gamefield& other):** Конструктор копирования, создает копию игрового поля.

**Gamefield(Gamefield&& other):** Конструктор перемещения, переносит данные из другого объекта.

**~Gamefield():** Деструктор, освобождает выделенную память.

**Cell& getCell(int x, int y):** Возвращает клетку по координатам.

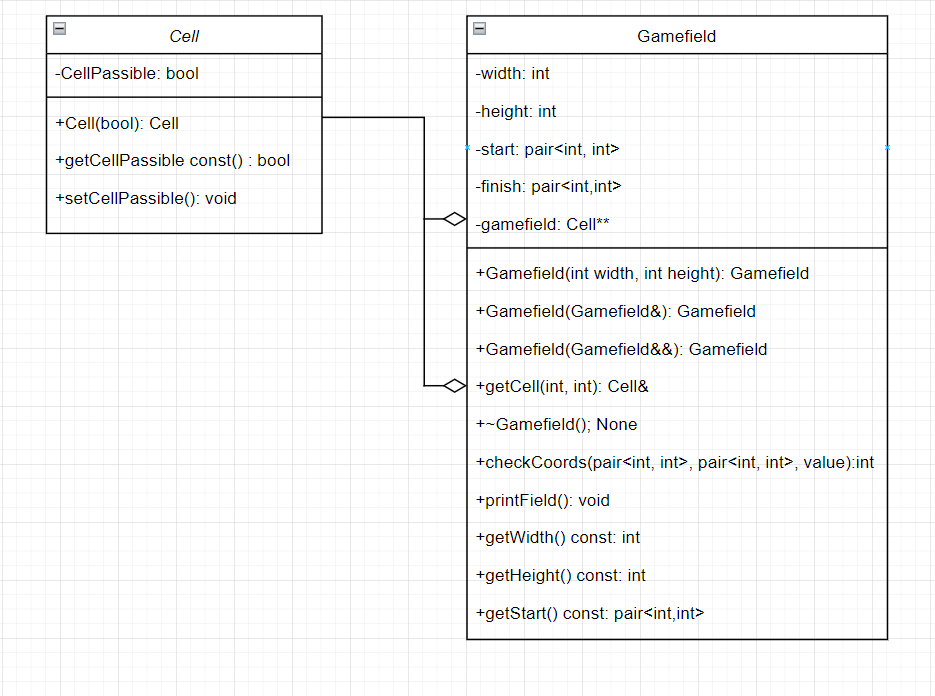
**void printField():** Выводит игровое поле на экран.

**int getWidth() const:** Возвращает ширину поля.

**int getHeight() const:** Возвращает высоту поля.

**std::pair<int, int> getStart() const:** Возвращает координаты начальной точки.

В классе *Playercontroller* обновлена функция *move*, которая теперь проверяет, может ли игрок переместиться в эту клетку



## Выводы

Был реализован класс поля и клетки. Реализованы конструкторы копирования, перемещения, а также деструктор.

# Приложение А Исходный код программы

Название файла: Cell.hpp

#ifndef cell\_hpp

#define cell\_hpp

class Cell

{

private:

    bool cellPassible;

public:

    Cell(bool cellPasible = true);

    bool getCellPassible() const;

    void setCellPassible(bool cellPassible);

};

#endif

Название файла: Cell.cpp

#include <iostream>

#include "cell.hpp"

Cell::Cell(bool cellPassible){

    this->cellPassible = cellPassible;

}

bool Cell::getCellPassible() const{

    return this->cellPassible;

}

void Cell::setCellPassible(bool cellPassible){

    this->cellPassible = cellPassible;

}

Название файла: gamefield.hpp

#ifndef gamefield\_hpp

#define gamefield\_hpp

#define D\_WIDTH 15

#define D\_HEIGHT 15

#include <iostream>

#include "Cell.hpp"

class Gamefield

{

private:

    Cell\*\* gamefield;

    int width, height;

    std::pair <int, int> start;

    std::pair <int, int> finish;

public:

    Gamefield(int width = D\_WIDTH, int height = D\_HEIGHT);

    Gamefield(Gamefield& other);

    Gamefield(Gamefield&& other);

    Cell& getCell(int x, int y);

    ~Gamefield();

    int checkCoords(std::pair<int,int> start,std::pair<int,int> finish ,int value) const;

    void printField();

    int getWidth() const;

    int getHeight() const;

    std::pair<int,int> getStart() const;

};

#endif

Название файла: Field.cpp

#include <iostream>

#include "gamefield.hpp"

#include "playercontroller.hpp"

Gamefield::Gamefield(int width, int height){

    if (width > 2 && width < 60 && height > 2 && height < 60) {

        this->width = width;

        this->height = height;

    }

    else{

        this->width = D\_WIDTH;

        this->height = D\_HEIGHT;

    }

    gamefield = new Cell\*[height];

    for(int i = 0; i < height; i++){

        gamefield[i] = new Cell[width];

        }

    if(checkCoords(start, finish, width) && checkCoords(start, finish, height)){

        this->start = start;

        this->finish = finish;

    }

    else{

        this->start = std::make\_pair(0,0);

        this->finish = std::make\_pair(D\_WIDTH - 1 , D\_HEIGHT - 1);

    }

}

Gamefield::Gamefield(Gamefield& other){

    width = other.width;

    height = other.height;

    start = other.start;

    finish = other.finish;

    gamefield = new Cell\*[height];

    for(int i = 0; i < height; i++){

        gamefield[i] = new Cell[width];

    }

    for(int i = 0; i < height; i++){

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

            gamefield[i][j] = other.gamefield[i][j];

        }

    }

}

Gamefield::Gamefield(Gamefield&& other) : width(other.width), height(other.height), start(std::move(other.start)), finish(std::move(finish)), gamefield(other.gamefield){

        other.gamefield = nullptr;

    }

Gamefield::~Gamefield(){

    for (int i = 0; i < height; i++)

    {

        delete [] gamefield[i];

    }

    delete [] gamefield;

}

Cell& Gamefield::getCell(int x, int y) {

    if (x >= 0 && x < width && y >= 0 && y < height) {

        return gamefield[x][y];

    } else {

        throw std::out\_of\_range("За поле вышел");

    }

}

void Gamefield::printField(){

    for(int i = 0; i < height; i++){

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

            if (i == start.second && j == start.first) {

                std::cout << 'S' << ' ';

            } else if (i == finish.second && j == finish.first) {

                std::cout << 'F' << ' ';

            } else {

                std::cout << (gamefield[i][j].getCellPassible() ? '1' : '0') << ' ';

            }

        }

        std::cout << '\n';

    }

}

int Gamefield::checkCoords(std::pair<int,int> start, std::pair<int,int> finish, int value) const {return start.first >= 0 && start.first < value && start.second >= 0 && start.second < value; }

int Gamefield::getWidth() const { return this->width; }

int Gamefield::getHeight() const { return this->height; }

std::pair<int,int> Gamefield::getStart() const{

    return start;

}