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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Объектно-ориентированное программирование»
Тема: Создание классов, конструкторов и методов классов

Студентка гр. 0382 _____ Тихонов С.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.
Освоить создание классов, конструкторов и методов в языке C++.

Задание.
Игровое поле представляет из себя прямоугольную плоскость разбитую на клетки. На поле на клетках в дальнейшем будут располагаться игрок, враги, элементы взаимодействия. Клетка может быть проходимой или непроходимой, в случае непроходимой клетки, на ней ничего не может располагаться. На поле должны быть две особые клетки: вход и выход. В дальнейшем игрок будет появляться на клетке входа, а затем выполнив определенный набор задач дойти до выхода.
Требования:

- Реализовать класс поля, который хранит набор клеток в виде двумерного массива.
- Реализовать класс клетки, которая хранит информацию о ее состоянии, а также того, что на ней находится.
- Создать интерфейс элемента клетки.
- Обеспечить появление клеток входа и выхода на поле. Данные клетки не должны быть появляться рядом.
- Для класса поля реализовать конструкторы копирования и перемещения, а также соответствующие операторы.
- Гарантировать отсутствие утечки памяти.

Выполнение работы.
В ходе выполнения лабораторной работы было реализовано 3 класса, а именно: Elem_Cell, Cell, Field.

Class Elem_Cell

Этот класс является интерфейсом другие классы будут от него наследоваться и иметь все его функции.

В классе созданы вспомогательные enum классы Elem и Type, в которых содержится информация о том, что находится на клетке и о типе клетки соответственно.

В классе объявлены приватные поля:

Elem _elem — элемент, находящийся на клетке.

T*ype _type* — тип клетки (начало/конец/стена/обычная).

Защищённые методы класса

```
virtual  void setElem(Elem elem)=0;
virtual void setType(Type type)=0;
virtual Elem getElem()=0;
virtual Type getType()=0;
virtual float Demage()=0;
virtual void getItam()=0;
```

Class Cell:

Это класс одной отдельной клетки поля. Он является дочерним для класса Elem_Cell и наследует его поля и методы.

Реализован конструктор класса *Cell(Elem elem , Type type)*. Принимает на вход то, что находится на ней и то, какого она типа.

Публичные методы класса:

void setElem(Elem elem) — метод, который позволяет установить элемент находящийся поверх клетки.

Elem getElem() — возвращает тип элемента находящегося на клетке.

void setType(Type type) — устанавливает тип клетки.

Type getType — возвращает тип клетки.

Class Field:

Это класс поля состоящий из массива клеток и некоторых параметров поля.

В классе объявлены публичные поля:

*Cell*** *cells* — двумерный массив ссылок на клетки.

int Width, Heigth — ширина и высота.

Реализован конструктор, который создает двумерный массив, содержащий в себе ссылки на клетки поля. Также реализованы конструкторы копирования и перемещения и соответствующие операторы для них.

Методы класса:

void inizializeion(); - начало заполнения

void creatEnterExit(); - создавание входа и выхода

void creatwall(); - создавание стен

bool setTypecell(int Width, int Height, Type type); - подготовка к установке определенного типа для клетке

bool setType_setelem(int Width, int Height,Elem elem, Type type); - установка нужного типа и элемента в клетку

void printField(); - вывод поля

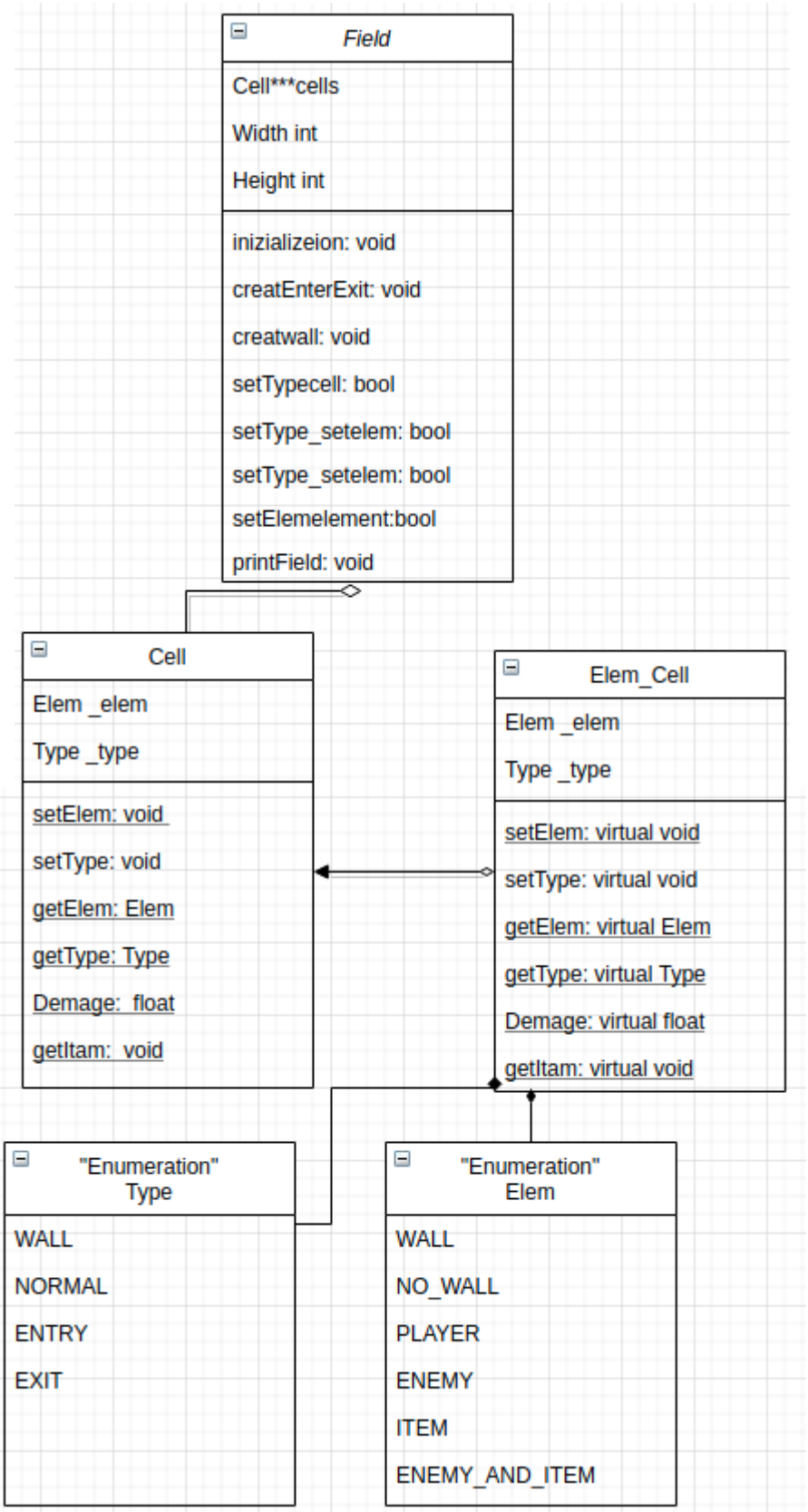
bool setElemelement(int W, int H, Elem elem); подготовка к установке нужного элемента в клетку

В классе реализован деструктор, в котором память, выделенная под двумерный массив, освобождается.

Тестирование.
Результат работы программы:
Входные данные (5,5)

```
0, 0    1, 2    0, 0    0, 0    0, 0
0, 0    1, 1    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    0, 0
0, 0    0, 0    0, 0    1, 3    0, 0
```

UML диаграмма



Выводы.

Было исследовано создание классов, методом и конструкторов на языке C++. Разработана программа, которая создает игровое поле и выводит его в консоль.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла - main.cpp

```
#include "Field.h"


int main()
{
   Field field(5,5);

   return 0;
}
```

Название файла — Field.h

```
#ifndef UNTITLED2_START_H
#define UNTITLED2_START_H
#include "Cells.h"
class Field{
public:
   Cell ***cells;
   int Height=10,Width=5;

   Field(int Width, int Height);
   Field(const Field& other);
   Field& operator=(const Field& other);
   Field& operator=(Field&& other);
   void inizializeion();
   void creatEnterExit();
   void creatwall();
   bool setTypecell(int Width, int Height, Type type);
   bool setType_setelem(int Width, int Height,Elem elem, Type type);
   bool setElemelement(int W, int H, Elem elem);
   void printField();
   ~Field();
};

#endif //UNTITLED2_START_H
```

Название файла — Field.cpp

```
#include "Field.h"
#include "Cells.h"
#include <iostream>

Field::Field(int width, int height): Height(height), Width(width) {
   cells=new Cell**[Width];
```

```
    for(int i =0; i<Width; i++){
        cells[i]=new Cell*[Height];
        for(int j=0; j<Height; j++){
            cells[i][j]=new Cell;
        }
    }
    if(Width<2 || Width>20 || Height<2 || Width>20){
        std::cout<<"Не верные данные";
        return;
    }
    for (int i=0; i<Width; i++){
        for (int j=0; j<Height; j++){
            cells[i][j]->setType(Type::NORMAL);
            cells[i][j]->setElem(Elem::NO_WALL);
        }
    }
    inizializeion();
    printField();
}

Field::Field(const Field& other){
    *this = other;
}

Field& Field::operator=(const Field& other) {
    this->Height = other.Height;
    this->Width = other.Width;
    for(int i = 0; i < this->Height; i++) {
        for (int j = 0; j < this->Width; j++) {
            this->cells[i][j] = other.cells[i][j];
        }
    }
    return *this;
}

Field& Field::operator=(Field&& other){
    this->Height = other.Height;
    other.Height = 0;
    this->Width = other.Width;
    other.Width = 0;
    for(int i = 0; i < this->Height; i++) {
        for (int j = 0; j < this->Width; j++) {
            this->cells[i][j] = other.cells[i][j];
        }
    }
    return *this;
```

```
};

Field::~Field() {
    for (int i = 0; i < Width; ++i) {
        for(int j=0; j<Height; j++){
            delete cells[i][j];
        }
        delete cells[i];
    }
    delete cells;
}
void Field::inizializeion() {
    creatwall();
    creatEnterExit();
}
void Field::creatwall() {
    for(int i=0; i<Width; i++){
        setTypecell(i,0,Type::WALL);
        setTypecell(i,Height-1,Type::WALL);
        setTypecell(0,i,Type::WALL);
        setTypecell(Width-1,i,Type::WALL);
        setElemelement(i,0,Elem::WALL);
        setElemelement(i,Height-1,Elem::WALL);
        setElemelement(0,i,Elem::WALL);
        setElemelement(Width-1,i,Elem::WALL);
    }
}

void Field::creatEnterExit(){
    setTypecell(0, 1, Type::ENTRY);
    setTypecell(Width-1,Height-2, Type::EXIT);
    setElemelement(0,1,Elem::NO_WALL);
    setElemelement(Width-1,Height-2,Elem::NO_WALL);
}

bool Field::setTypecell(int W, int H, Type type){
    return setType_setelem(W,H,cells[W][H]->getElem(),type);
}

bool Field::setElemelement(int W, int H, Elem elem){
    return setType_setelem(W,H,elem, cells[W][H]->getType());
}
///////создание входа выхода и хз

bool Field::setType_setelem(int width, int height, Elem elem, Type type) {
    if ((width > Width) || (height > Height)) {
```

```
        return false;
    }

    cells[width][height]->setElem(elem);
    cells[width][height]->setType(type);


    return true;
}
void Field::printField() {
    for (int i = 0; i < Width; i++){
        for (int j = 0; j < Height; j++){
            std::cout << static_cast<unsigned short>(cells[i][j]->getElem()) << ", "
                    << static_cast<unsigned short>(cells[i][j]->getType()) << '\t'; //
Преобразование типов.
        }

        std::cout << std::endl;
    }

    std::cout << std::endl;
}
```

Название файла — Cells.h

```
//#include "Field.h"
#ifndef UNTITLED2_CELLS_H
#define UNTITLED2_CELLS_H


enum class Elem : unsigned short {
    WALL,
    NO_WALL,
    PLAYER,
    ENEMY,
    ITEM,
    ENEMY_AND_ITEM
};

enum class Type : unsigned short {
    WALL,
    NORMAL,
    ENTRY,
    EXIT
};
```

```
class Elem_Cell{
protected:
    Elem _elem;
    Type _type;
private:
    virtual  void setElem(Elem elem)=0;
    virtual void setType(Type type)=0;
    virtual Elem getElem()=0;
    virtual Type getType()=0;
    virtual float Demage()=0;
    virtual void getItam()=0;
};



class Cell: public Elem_Cell{
protected:
    //Elem _elem;
    //Type _type;
public:
    Cell(Elem elem, Type type);
    Cell();
    void setElem(Elem elem) override;
    void setType(Type type) override;
    Elem getElem() override;
    Type getType() override;
    void getItam() override;
    float Demage() override;
    Cell(const Cell&) = default;
    Cell(Cell&&) = default;
    ~Cell() = default;
};


#endif //UNTITLED2_CELLS_H
```

Название файла — Cells.cpp

```
//#include "Field.h"
#ifndef UNTITLED2_CELLS_H
#define UNTITLED2_CELLS_H


enum class Elem : unsigned short {
    WALL,
    NO_WALL,
    PLAYER,
```

```
    ENEMY,
    ITEM,
    ENEMY_AND_ITEM
};

enum class Type : unsigned short {
    WALL,
    NORMAL,
    ENTRY,
    EXIT
};

class Elem_Cell{
protected:
    Elem _elem;
    Type _type;
private:
    virtual  void setElem(Elem elem)=0;
    virtual void setType(Type type)=0;
    virtual Elem getElem()=0;
    virtual Type getType()=0;
    virtual float Demage()=0;
    virtual void getItam()=0;
};



class Cell: public Elem_Cell{
protected:
    //Elem _elem;
    //Type _type;
public:
    Cell(Elem elem, Type type);
    Cell();
    void setElem(Elem elem) override;
    void setType(Type type) override;
    Elem getElem() override;
    Type getType() override;
    void getItam() override;
    float Demage() override;
    Cell(const Cell&) = default;
    Cell(Cell&&) = default;
    ~Cell() = default;
};
```

```
#endif //UNTITLED2_CELLS_H
```