**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 классов, методов классов, наследование.**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент гр. 8303 | |  | Кибардин А.Б. |
| Преподаватель |  |  | Филатов Ан.Ю. |

Санкт-Петербург

2020

**Цель работы.**

Научиться создавать классы и их конструкторы, реализовать методы классов и познакомиться с наследованием классов.

**Задание.**

Разработать и реализовать набор классов:

* Класс игрового поля
* Набор классов юнитов

Игровое поле является контейнером для объектов представляющим прямоугольную сетку. Основные требования к классу игрового поля:

* Создание поля произвольного размера
* Контроль максимального количества объектов на поле
* Возможность добавления и удаления объектов на поле
* Возможность копирования поля (включая объекты на нем)
* Для хранения запрещается использовать контейнеры из stl

Юнит является объектом, размещаемым на поля боя. Один юнит представляет собой отряд. Основные требования к классам юнитов:

* Все юниты должны иметь как минимум один общий интерфейс
* Реализованы 3 типа юнитов (например, пехота, лучники, конница)
* Реализованы 2 вида юнитов для каждого типа(например, для пехоты могут быть созданы мечники и копейщики)
* Юниты имеют характеристики, отражающие их основные атрибуты, такие как здоровье, броня, атака.
* Юнит имеет возможность перемещаться по карте

**Ход работы.**

Разработаны и реализованы классы:

- Класс игрового поля Field

- Набор классов юнитов: dwarf, paladin, elf, bard, flamen, necromanser

Класс игрового поля Field является контейнером для объектов представляющим собой прямоугольную сетку. Создано поле произвольного размера. Для работы с полем созданы методы удаления и добавления юнитов, смены координат юнитов, проверки корректности координат, вывода поля, конструкторы и операторы копирования и присваивания.

Все классы юнитов имеют один общий интерфейс, описанный в базовом классе Unit. Созданы три типа юнитов — маг, воин, вор. Для каждого типа создано два подтипа. Каждый юнит имеет возможность перемещаться по полю.

**Выводы.**

В ходе выполнения работы были разработаны и реализованы классы игрового поля и набор классов юнитов. Так же получены навыки по созданию конструкторов, методов классов и их наследованию.

**Приложение А.**

**Исходный код.**

Файл bard.cpp:

#include "Bard.h"

Bard::Bard(){

health = 125;

armor = 0;

attack = 5;

attackRange = 10;

}

char Bard::getClass() const {

return 'B';

}

Файл bard.h:

#ifndef OOP\_1\_BARD\_H

#define OOP\_1\_BARD\_H

#include "Thief.h"

class Bard: public Thief {

public:

Bard();

char getClass() const;

};

#endif *//OOP\_1\_BARD\_H*

Файл cell.cpp:

#include "Cell.h"

Cell::Cell():Object(nullptr){}

void Cell::clearCell() {

Object = nullptr;

}

void Cell::setCharacter(Unit\* newUnit) {

if(Object == nullptr)

Object = newUnit;

}

Unit\* Cell::getCharacter() const{

return this->Object;

}

Cell::Cell(const Cell& obj):Cell()

{

this->Object = obj.Object;

}

Cell& Cell::operator=(const Cell& obj) {

if(this == &obj)

return \*this;

this->Object = obj.Object;

return \*this;

}

Файл cell.h:

#ifndef OOP\_1\_CELL\_H

#define OOP\_1\_CELL\_H

#include "Unit.h"

class Unit;

class Cell {

Unit\* Object;

public:

Cell();

Cell(const Cell& obj);

Cell& operator=(const Cell& obj);

void clearCell();

void setCharacter(Unit\* Object);

Unit\* getCharacter() const;

};

#endif *//OOP\_1\_CELL\_H*

Файл dwarf.cpp:

#include "Dwarf.h"

Dwarf::Dwarf(){

health = 250;

armor = 250;

attack = 40;

attackRange = 2;

}

char Dwarf::getClass() const {

return 'D';

}

Файл dwarf.h:

#ifndef OOP\_1\_DWARF\_H

#define OOP\_1\_DWARF\_H

#include "Warrior.h"

class Dwarf: public Warrior{

public:

Dwarf();

char getClass() const;

};

#endif *//OOP\_1\_DWARF\_H*

Файл elf.cpp:

#include "Elf.h"

Elf::Elf() {

health = 125;

armor = 50;

attack = 30;

attackRange = 2;

}

char Elf::getClass() const {

return 'E';

}

Файл elf.h:

#ifndef OOP\_1\_ELF\_H

#define OOP\_1\_ELF\_H

#include "Thief.h"

class Elf: public Thief {

public:

Elf();

char getClass() const;

};

#endif *//OOP\_1\_ELF\_H*

Файл field.cpp:

#include <iostream>

#include "Field.h"

Field::Field(int x, int y) :X(x), Y(y), matrix(new Cell\*[X]), objectsCounter(0), maxObjectsOnField(x\*y){

for(int i = 0; i < X; i++) {

matrix[i] = new Cell[Y];

for(int j = 0; j < Y; j++)

matrix[i][j].setCharacter(nullptr);

}

}

Field::Field():Field(0,0) {}

Field::~Field()

{

if(matrix)

{

for(int i = 0; i < X; i++)

if(matrix + i)

delete[] matrix[i];

delete[] matrix;

}

}

void Field::addObject(Unit \*Object, int x, int y) {

if(!Object->isOnField())

Object->replace(this);

if(checkPoint(x,y) && !matrix[x][y].getCharacter() && objectsCounter < maxObjectsOnField) {

matrix[x][y].setCharacter(Object);

Object->setX(x);

Object->setY(y);

objectsCounter++;

}

}

int Field::getObjectCounter() const{

return objectsCounter;

}

void Field::deleteObject(Unit \*Object) {

int requiredX = Object->getX();

int requiredY = Object->getY();

if(checkPoint(requiredX, requiredY) && matrix[requiredX][requiredY].getCharacter() == Object) {

matrix[requiredX][requiredY].clearCell();

}

objectsCounter--;

}

void Field::print() const{

printf("------------------\n");

for(int i = 0; i < X;i++){

for(int j = 0; j < Y; j++)

if(matrix[i][j].getCharacter() != nullptr)

std::cout << matrix[i][j].getCharacter()->getClass() << " ";

else

std::cout << ". ";

std::cout << std::endl;

}

printf("------------------\n");

}

bool Field::checkPoint(int x, int y) const {

return x < X && y < Y;

}

void Field::swapCharacters(int x1, int y1, int x2, int y2) {

if(x1 == x2 && y1 == y2)

return;

*//std::swap(matrix[x1][y1].Object, matrix[x2][y2].Object);*

Unit\* tmp = matrix[x1][y1].getCharacter();

matrix[x1][y1].clearCell();

matrix[x1][y1].setCharacter(matrix[x2][y2].getCharacter());

matrix[x2][y2].clearCell();

matrix[x2][y2].setCharacter(tmp);

}

int Field::getX() const {

if(matrix != nullptr)

return X;

return -1;

}

int Field::getY() const {

if(matrix != nullptr)

return Y;

return -1;

}

Field::Field(const Field& obj) : Field(obj.X,obj.Y)

{

objectsCounter = obj.objectsCounter;

for(int i = 0; i < X; i++)

for(int j = 0; j < Y; j++)

this->matrix[X][Y] = obj.matrix[X][Y];

}

Field::Field(Field&& obj):X(0), Y(0), matrix(nullptr){

std::swap(matrix, obj.matrix);

std::swap(X, obj.X);

std::swap(Y, obj.Y);

std::swap(objectsCounter, obj.objectsCounter);

std::swap(maxObjectsOnField, obj.maxObjectsOnField);

}

Field& Field::operator=(const Field& obj) {

if(this == &obj)

return \*this;

this->matrix = new Cell\*[obj.X];

this->X = obj.X;

this->Y = obj.Y;

this->objectsCounter = obj.objectsCounter;

this->maxObjectsOnField = obj.maxObjectsOnField;

for(int i = 0; i < X; i++) {

this->matrix[i] = new Cell[Y];

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

this->matrix[i][j] = obj.matrix[i][j];

}

}

return \*this;

}

Field& Field::operator=(Field &&obj) {

if(this == &obj)

return \*this;

std::swap(matrix, obj.matrix);

std::swap(X, obj.X);

std::swap(Y, obj.Y);

std::swap(objectsCounter, obj.objectsCounter);

std::swap(maxObjectsOnField, obj.maxObjectsOnField);

return \*this;

}

Файл field.h:

#ifndef OOP\_1\_FIELD\_H

#define OOP\_1\_FIELD\_H

#include "Cell.h"

#include "Unit.h"

class Cell;

class Unit;

class Field{

int X, Y;

Cell\*\* matrix;

int objectsCounter;

int maxObjectsOnField;

public:

Field();

Field(int x, int y);

~Field();

int getX() const;

int getY() const;

bool checkPoint(int x, int y) const;

void swapCharacters(int x1, int y1, int x2, int y2);

int getObjectCounter() const;

void addObject(Unit\* Object, int x, int y);

void deleteObject(Unit\* Object);

void print() const;

Field(const Field& obj); *// copy constructor*

Field(Field&& obj); *// move constructor*

Field& operator=(const Field& obj);

Field& operator=(Field&& obj);

};

#endif *//OOP\_1\_FIELD\_H*

Файл flamen.cpp:

#include "Flamen.h"

Flamen::Flamen(){

health = 125;

armor = 0;

attack = 15;

attackRange = 5;

}

char Flamen::getClass() const {

return 'F';

}

Файл flamen.h:

#ifndef OOP\_1\_FLAMEN\_H

#define OOP\_1\_FLAMEN\_H

#include "Wizard.h"

class Flamen: public Wizard {

public:

Flamen();

char getClass() const;

};

#endif *//OOP\_1\_FLAMEN\_H*

Файл main.cpp:

#include <iostream>

#include "Field.h"

#include "Dwarf.h"

#include "Paladin.h"

#include "Elf.h"

#include "Bard.h"

#include "Flamen.h"

#include "Necromancer.h"

int main() {

puts("first example");

puts("created first field");

Field gameField(5,8);

gameField.print();

printf("add dwarf on field\n");

Dwarf zoltan;

gameField.addObject(&zoltan, 2, 2);

gameField.print();

printf("move dwarf\n");

zoltan.move(3,5);

gameField.print();

printf("delete dwarf\n");

zoltan.deleteFromField();

gameField.print();

puts("\nsecond example");

Elf legolas;

puts("add elf on field");

gameField.addObject(&legolas, 1, 1);

gameField.print();

puts("move first field to second");

Field newField(std::move(gameField));

puts("second field");

newField.print();

puts("first field");

gameField.print();

puts("\nthird example");

puts("create third field");

Field third(5,5);

puts("copy third field to first");

gameField = third;

puts("third field");

third.print();

puts("first field");

gameField.print();

puts("add dwarf to third field");

third.addObject(&zoltan, 1, 1);

third.print();

puts("first field");

gameField.print();

return 0;

}

Файл necromancer.cpp:

#include "Necromancer.h"

Necromancer::Necromancer(){

health = 100;

armor = 25;

attack = 10;

attackRange = 5;

}

char Necromancer::getClass() const {

return 'N';

}

Файл necromancer.h:

#ifndef OOP\_1\_NECROMANCER\_H

#define OOP\_1\_NECROMANCER\_H

#include "Wizard.h"

class Necromancer: public Wizard {

public:

Necromancer();

char getClass() const;

};

#endif *//OOP\_1\_NECROMANCER\_H*

Файл paladin.cpp:

#include "Paladin.h"

Paladin::Paladin(){

health = 300;

armor = 150;

attack = 35;

attackRange = 2;

}

char Paladin::getClass() const {

return 'P';

}

Файл paladin.h:

#ifndef OOP\_1\_PALADIN\_H

#define OOP\_1\_PALADIN\_H

#include "Warrior.h"

class Paladin: public Warrior{

public:

Paladin();

char getClass() const;

};

#endif *//OOP\_1\_PALADIN\_H*

Файл thief.cpp:

#include "Thief.h"

Файл thief.h:

#ifndef OOP\_1\_THIEF\_H

#define OOP\_1\_THIEF\_H

#include "Unit.h"

class Thief: public Unit{

friend class Bard;

friend class Elf;

};

#endif *//OOP\_1\_THIEF\_H*

Файл unit.cpp:

#include <cstdio>

#include "Unit.h"

Unit::Unit():health(200), armor(200), attack(20), attackRange(2), gameField(nullptr)

{}

int Unit::getX() const {

if(gameField != nullptr)

return this->x;

return -1;

}

int Unit::getY() const {

if(gameField != nullptr)

return this->y;

return -1;

}

void Unit::setX(int x) {

this->x = x;

}

void Unit::setY(int y) {

this->y = y;

}

void Unit::move(int X, int Y){

if(gameField != nullptr && gameField->checkPoint(X, Y))

{

printf("start coord:%d %d\n", this->x, this->y);

gameField->swapCharacters(this->x,this->y, X, Y);

this->y = Y;

this->x = X;

}

}

bool Unit::isOnField() {

return gameField != nullptr;

}

void Unit::deleteFromField() {

gameField->deleteObject(this);

gameField = nullptr;

}

void Unit::replace(Field \*gameField) {

this->gameField = gameField;

}

Файл unit.h:

#ifndef OOP\_1\_UNIT\_H

#define OOP\_1\_UNIT\_H

#include "Cell.h"

#include "Field.h"

class Field;

class Cell;

class Unit {

int health;

int armor;

int attack;

int attackRange;

Field\* gameField;

int x;

int y;

public:

Unit();

bool isOnField();

int getX() const;

int getY() const;

void setX(int x);

void setY(int y);

void move(int X, int Y);

void replace(Field\* gameField);

virtual char getClass() const = 0 ;

void deleteFromField();

friend class Wizard;

friend class Warrior;

friend class Thief;

friend class Elf;

friend class Bard;

friend class Dwarf;

friend class Paladin;

friend class Flamen;

friend class Necromancer;

};

#endif *//OOP\_1\_UNIT\_H*

Файл warrior.cpp:

#include "Warrior.h"

Файл warrior.h:

#ifndef OOP\_1\_WARRIOR\_H

#define OOP\_1\_WARRIOR\_H

#include "Unit.h"

class Warrior: public Unit {

public:

friend class Dwarf;

friend class Paladin;

};

#endif *//OOP\_1\_WARRIOR\_H*

Файл wizard.cpp:

#include "Wizard.h"

char Wizard::getClass() const {

return 'W';

}

Файл wizard.h:

#ifndef OOP\_1\_WIZARD\_H

#define OOP\_1\_WIZARD\_H

#include "Unit.h"

class Wizard: public Unit {

public:

char getClass() const;

friend class Flamen;

friend class Necromancer;

};

#endif *//OOP\_1\_WIZARD\_H*