**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: «Консольная игра»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6382 |  | Черкасова Е.И. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2017

**Цель работы.**

Реализовать класс Объект, а также класс Поле\_боя, являющийся обёрткой над контейнерами-армиями, хранящим экземпляры класса Объект.

**Постановка задачи.**

Объект должен иметь:

1. Двумерные координаты.
2. Очки жизней.
3. Говорящий конструктор, говорящий деструктор.
4. Функцию, которая принимает урон.
5. Функцию, отвечающую на вопрос, есть ли Объект на указанной позиции.
6. Считываться из файла.

Поле\_боя должно иметь:

1. Контейнеры, хранящие объекты.
2. Отрисовку Поля\_боя на экране (разными цветами).
3. Загружать объекты из файла
4. Определять, располагается ли на заданных координатах объект той или иной армии.

**Дополнения:**

1. Обновить класс Объект таким образом, чтобы в нём вёлся учёт созданных объектов. Так, чтобы каждому новому создаваемому объекту присваивался новый идентификатор. Добавить “корону” - индикатор, что ещё существует хотя бы один объект. Добавить перегрузку операторов << и >> для Объекта и для Поля боя. Оформить циклы for в с++11 стиле (с использованием auto).

**Реализация.**

Для решения задачи используются классы Oblect и Field. Возможен ввод данных как из файла (input), так и, исрользуя консоль (причем при консольном вводе, данные также заносятся в файл (fromcondole)). Имееются говорящие конструктор и деструктор в обоих классах.

Объекты хранятся в контейнерах. Указатели на данные контейнеры лежат в векторе contains (индекс - номер армии). Полее боя map представляется двумерным массивом типа char. После нанесения урона соответсвующие изменения отображаются в карте. По координатам возможно узнать, к какой армии принадлежит объект, его состояние и др.

Учет объектов ведется с помочью статической переменной (счетчика), находящейся в поле класса Object (count\_ob) и записывается в поле id р при создании очередного объекта как его идинтификато.

Когда все объекты уничтожены, корона тоже уничтожается, что ведёт к поражению в игре. Корона храниться как shared\_ptr в членах данных объектов и задается сразу при создании объекта (const shared\_ptr<>).

Для циклов foreach (for c++11) реализован итератор (begin, end), сделаны соответсвующие перегрузки операторов (!=, =) .

**Класс Oblect:**

class Object {

int x;

int y;

int hp;

public:

Object(int x\_, int y\_, int hp\_);

Object(fstream &f) ;

// destructor ~Elem (see file list)

int getx(){return x;}

int gety(){return y;}

int gethp(){return hp;}

void print\_ob() ;

void print\_ob\_simple();

void read\_obj(fstream &f);

void damage(int d);

bool is\_obj(int i, int j, char \*\*m);

};

**Класс Field:**

class Field{

friend class Object;

friend class List<Object>;

int x\_size;

int y\_size;

int army\_amount;

char \*\*map; *//map[y][x] - access*

vector <List<Object>\*> all\_cont;

public:

Field( int x\_size\_ = 8, int y\_size\_ = 8, int army\_ = 2);

~Field();

char \*\*getmap(){return map;}

vector <List<Object>\*> getconts() {return all\_cont;}

void print\_field();

void print\_cont();

void add\_to\_map(int i, int j);

void read\_file(fstream &f);

int find\_ob\_index(int i, int j);

int belonging\_army(int i, int j);

void check\_map();

};

**Класс Crown:**

class Crown{

const string color;

public:

Crown(string color);

~Crown();

const string &getcolor() const;

};

**Вывод.**

В результате выполнения лабораторной работы была создана имитация консольной игры, использовался собственный шаблонный контейнер с внешним итератором, перегружены необходимые функции. Реализованы необходимые требования к объектам.