**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 6382 |  | Мартыненко П.П. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2018

**Цель работы.**

Изучить концепции объектно-ориентированного программирования на примере реализации консольной игры.

**Формулировка задания.**

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

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.

**Пример работы программы.**

Входной файл:

10 10

4

0 0 80

1 1 100

2 2 100

0 2 100

3

9 9 100

9 8 100

7 5 100

*Результат работы:*

Field:

    x\_size = 10

    y\_size = 10

Object:

     x  = 0

     y  = 0

     hp = 80

Object.

Object:

     x  = 1

     y  = 1

     hp = 100

Object.

Object:

     x  = 2

     y  = 2

     hp = 100

Object.

Object:

     x  = 0

     y  = 2

     hp = 100

Object.

Object:

     x  = 9

     y  = 9

     hp = 100

Object.

Object:

     x  = 9

     y  = 8

     hp = 100

Object.

Object:

     x  = 7

     y  = 5

     hp = 100

Object.

Field.

 0123456789

+----------+

0|o.........|

1|.o........|

2|o.o.......|

3|..........|

4|..........|

5|.......o..|

6|..........|

7|..........|

8|.........o|

9|.........o|

+----------+

input coordinates of target (separated by space) [q to exit]: 0 0

Selected target is red

input a damage value: 100

 0123456789

+----------+

0|x.........|

1|.o........|

2|o.o.......|

3|..........|

4|..........|

5|.......o..|

6|..........|

7|..........|

8|.........o|

9|.........o|

+----------+

input coordinates of target (separated by space) [q to exit]: q

 0123456789

+----------+

0|x.........|

1|.o........|

2|o.o.......|

3|..........|

4|..........|

5|.......o..|

6|..........|

7|..........|

8|.........o|

9|.........o|

+----------+

~Field:

~Object:

     x  = 0

     y  = 0

     hp = -20

~Object.

~Object:

     x  = 1

     y  = 1

     hp = 100

~Object.

~Object:

     x  = 2

     y  = 2

     hp = 100

~Object.

~Object:

     x  = 0

     y  = 2

     hp = 100

~Object.

~Object:

     x  = 9

     y  = 9

     hp = 100

~Object.

~Object:

     x  = 9

     y  = 8

     hp = 100

~Object.

~Object:

     x  = 7

     y  = 5

     hp = 100

~Object.

~Field.

**Основные теоретические положения.**

В качестве контейнеров, хранящих объекты, выступают двусвязные списки, используемые в предыдущей лабораторной работе.

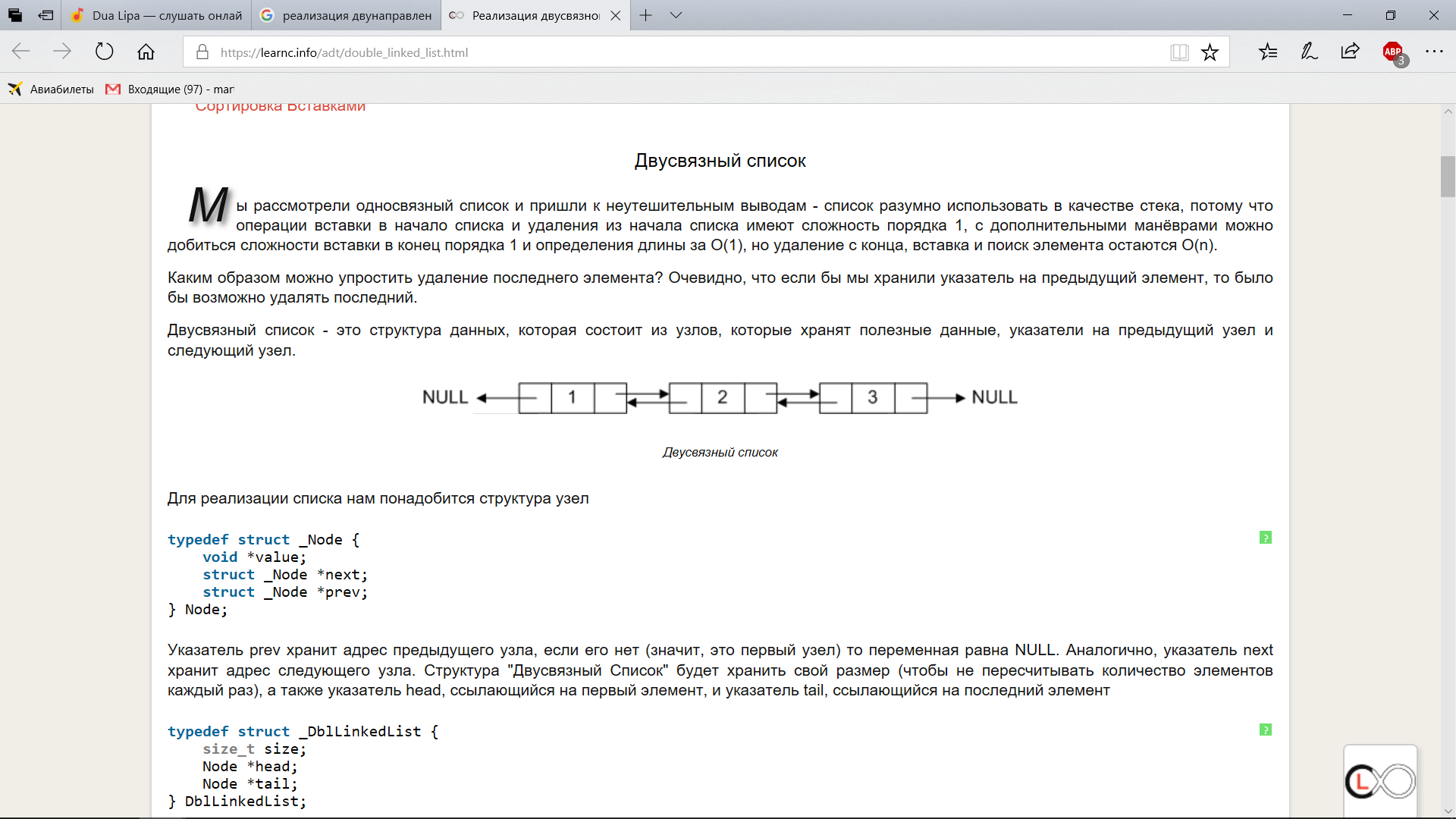
Двусвязный список - это структура данных, которая состоит из узлов, которые хранят полезные данные, указатели на предыдущий узел и следующий узел.

Рис.1.Двусвязный список.

**Спецификация программы.**

Для класса Object определены следующие операции:

1. Говорящие конструктор и деструктор – Object(int x, int y, int hp) и ~Object() соответственно.
2. Функция, которая принимает урон – int damage(int dmg).
3. Функция, отвечающая на вопрос, есть ли Объект на указанной позиции – bool isObject(int x1, int y1).
4. Считывание из файла – Object(ifstream& f);
5. Функция, отвечающая на вопрос, «убит» ли объект – bool isDead().

Для класса Field определены следующие операции:

1. Говорящие конструктор и деструктор – Field(ifstream &f) и ~Field() соответственно.
2. Функция, атакующая объект на заданной позиции – void attack(int x,int y,int dmg).
3. Отрисовка поля боя на экране (разными цветами) – void paint().
4. Загружать объекты из файла – void getArmies(ifstream& f).
5. Определять, располагается ли на заданных координатах объект той или иной армии – bool isIt(int x1, int y1).

Для класса Iterator определены следующие операции:

* Обращение к первому элементу списка. Возвращает класс Iterator, указывающий на первый элемент – метод begin().
* Обращение к «последнему» элементу списка. Возвращает класс Iterator, указывающий на нулевой элемент – метод end(). Используется для обозначения конца списка.
* Обращение к последнему элементу списка. Возвращает класс Iterator, указывающий на последний элемент – метод last().
* Разыменовывание. Возвращает ссылку на элемент, хранящийся в узле – **operator \***,
* Инкремент. Переход к следующему элементу (берет из узла и сохраняет адрес следующего узла) – **operator ++**,
* Декремент. Переход к предыдущему элементу (берет из узла и сохраняет адрес предыдущего узла) – **operator --**,
* Равенство и неравенство (простое сравнение указателей) – **operator ==** & **operator !=** ,
* Переход к узлу, на который указывает Iterator– **operator ->.**

Для класса List определены следующие операции:

* Вставка в начало и конец списка, а также вставка по номеру – методы pushFront(…), pushBack(…) и Insert(…) соответственно.
* Удаление из начала и конца списка, а также удаление по номеру – методы popFront(…), popBack(…) и Delete(…) соответственно.
* Показ списка – метод show().

**Выводы.**

В данной лабораторной работе были рассмотрены и изучены основные концепции объектно-ориентированного программирования, такие как дружественные классы, перегрузка операторов, шаблоны классов и т.д. Также был реализована консольная игра в рамках ООП.