**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 |  | Камалеев Р.М. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2018

**Цель работы.**

Изучение концепции ООП на примере реализации консольной игры.

**Постановка задачи.**

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.

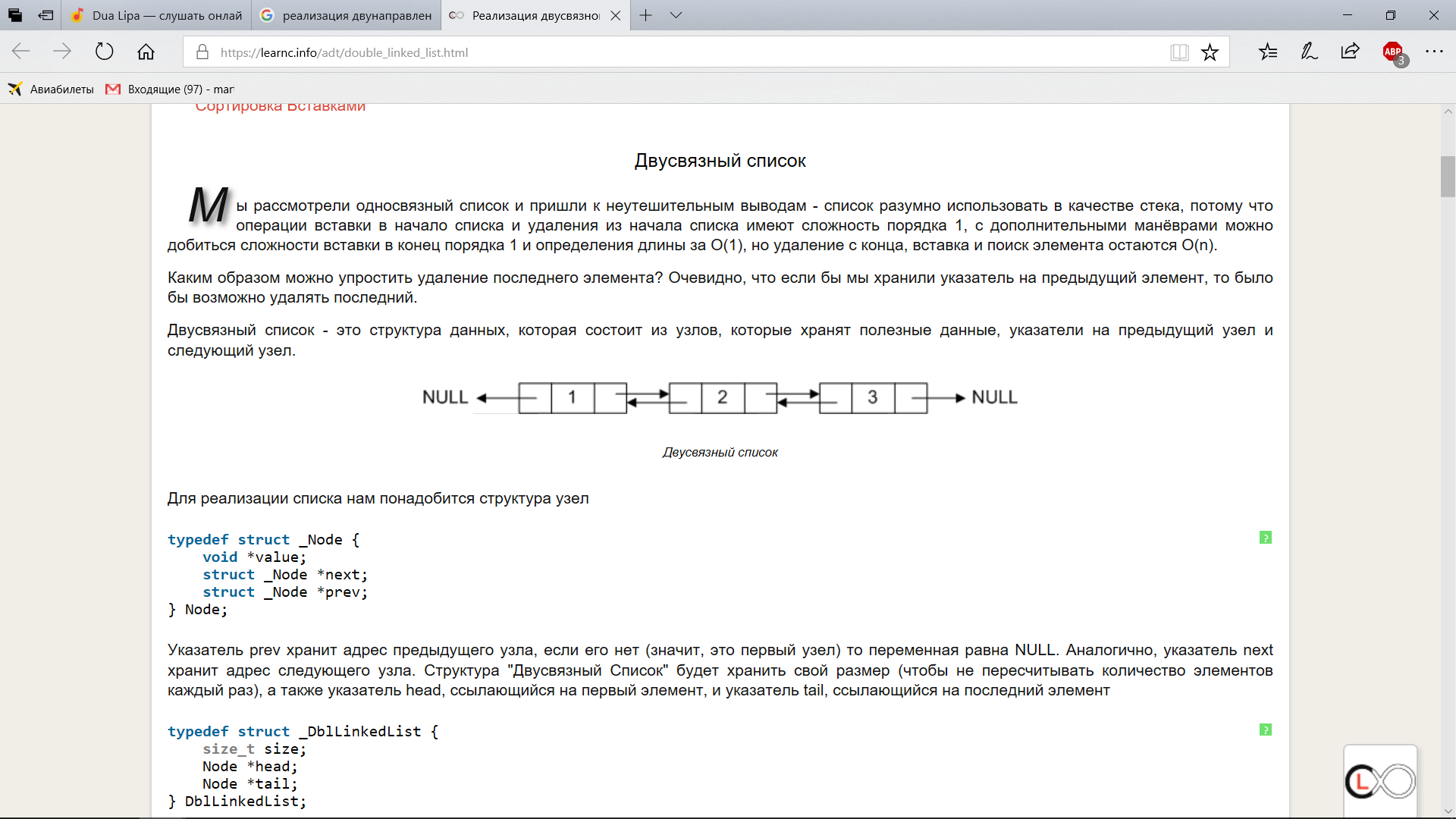


Рисунок 1. Двусвязный список.

**Спецификация программы.**

* Для класса Object определены следующие операции:

1. Говорящие конструктор и деструктор – Object(ifstream& fl) и ~Object() соответственно.
2. Функция, определяющая, есть ли объект на указанной позиции – bool Is\_Located(int x1, int y1).
3. Функция, принимающая урон – int Damage(int damage).
4. Функция, определяющая, убит ли объект – bool Is\_Dead().

* Для класса Field определены следующие операции:

1. Говорящие конструктор и деструктор – Field(ifstream &fl) и ~Field() соответственно.
2. Функция, инициализирующая объекты-контейнеры из файла – void Get\_Armies(ifstream& fl).
3. Функция, атакующая объект на заданной позиции – void Attack(int x, int y, int damage).
4. Функция, выполняющая отрисовку поля боя разными цветами – void Arena().
5. Функция, определяющая, куда выполнено попадание (по объекту красной армии, зелёной армии или мимо) – bool Is\_Target\_Hit(int x1, int y1).

* Для класса Iterator определены следующие операции:

1. T& operator \*() - оператор разыменования
2. T\* operator ->() - оператор доступа к полям по указателю
3. T operator [](int index) - оператор доступа к полям по индексу
4. Iterator operator ++(int) - оператор-инкремент
5. Iterator operator --(int) - оператор-декремент
6. friend bool operator !=(const Iterator& x, const Iterator& y) - оператор равенства
7. friend bool operator ==(const Iterator& x, const Iterator& y) - оператор неравенства

* Для класса List определены следующие операции:

1. ~List() – деструктор
2. void Print() – выводит на экран список
3. void Supl\_End(T info) – добавляет элемент в конец списка
4. void Supl\_Begin(T info) – добавляет элемент в начало списка
5. void Supl\_Pos(T info, int pos) – добавляет элемент по позиции
6. void Erase\_End(T info) – удаляет элемент из конца списка
7. void Erase\_Begin(T info) – удаляет элемент из начала списка
8. void Erase\_Pos(T info, int pos) – удаляет элемент по позиции

**Выводы.**

В ходе данной лабораторной работы были рассмотрены и изучены основные концепции объектно-ориентированного программирования (ООП), такие как дружественные классы, перегрузка операторов, шаблоны классов, вызов конструкторов и деструкторов. С использованием всех этих принципов была реализована консольная игра.