**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**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.**

Дополнить проект, реализованный в первой лабораторной работе, следующим образом:

* 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. Когда все объекты уничтожены, корона тоже уничтожается, что ведёт к поражению в игре. Корона должна храниться как shared\_ptr в членах данных объектов и задаваться сразу при создании объекта (const shared\_ptr<>).
* добавить перегрузку операторов << и >> для Объекта и для Поля боя.
* оформить циклы for в с++11 стиле (с использованием auto).

**Основные теоретические положения.**

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

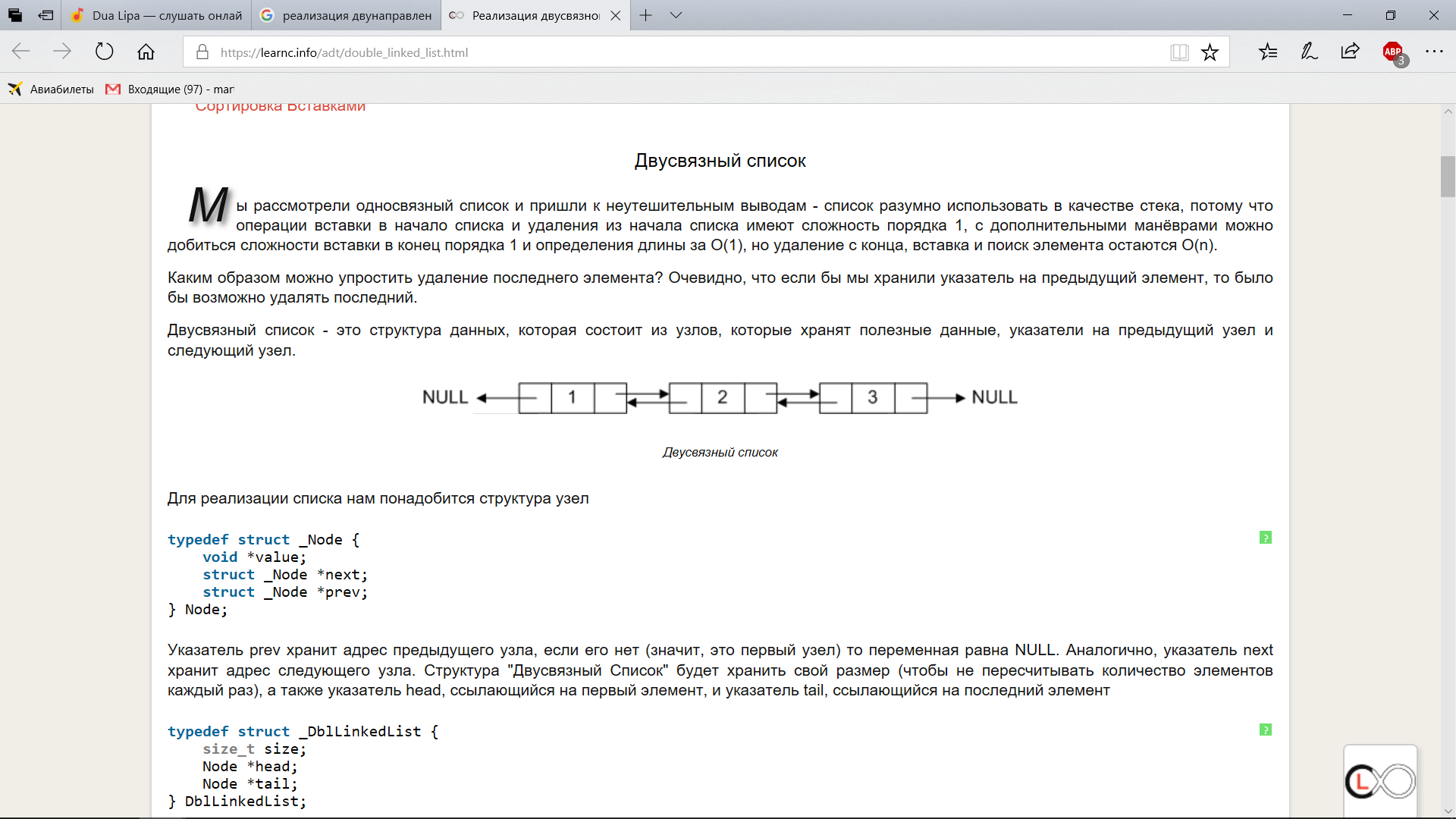
Двусвязный список - это структура данных, которая состоит из узлов, которые хранят полезные данные, указатели на предыдущий узел и следующий узел.

Рис.1.Двусвязный список.

**Спецификация программы.**

Для класса Object определены следующие операции:

1. Говорящие конструктор и деструктор – Object(int x, int y, int hp, const shared\_ptr<Crown> crown) 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, const shared\_ptr<Crown>& clr);
5. Функция, отвечающая на вопрос, «убит» ли объект – bool isDead().
6. Перегруженные операторы >> и <<, производящие ввод и вывод из потока ввода/вывода соответственно.

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 getA(ifstream& f, const shared\_ptr<Crown>& clr) и void getB(ifstream& f, const shared\_ptr<Crown>& clr).
5. Определять, располагается ли на заданных координатах объект той или иной армии – bool isIt(int x1, int y1).
6. Перегруженные операторы >> и <<, производящие ввод и вывод из потока ввода/вывода соответственно.

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.