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

Санкт-Петербург

2018

**Цель работы.**

Изучение концепции ООП на примере реализации консольной игры. Получение навыков работы с умными указателями.

**Постановка задачи.**

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).

**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

Crown:

color = red

Crown.

Object:

x = 0

y = 0

hp = 80

crown = red

id = 0

Object.

Object:

x = 1

y = 1

hp = 100

crown = red

id = 1

Object.

Object:

x = 2

y = 2

hp = 100

crown = red

id = 2

Object.

Object:

x = 0

y = 2

hp = 100

crown = red

id = 3

Object.

Crown:

color = green

Crown.

Object:

x = 9

y = 9

hp = 100

crown = green

id = 4

Object.

Object:

x = 9

y = 8

hp = 100

crown = green

id = 5

Object.

Object:

x = 7

y = 5

hp = 100

crown = green

id = 6

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|..........|

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

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

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

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

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

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

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

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

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

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

~Object:

x = 0

y = 0

hp = -20

crown = red

id = 0

~Object.

input coordinates of target (separated by space) [q to exit]: 1 1

Selected target is red

input a damage value: 100

0123456789

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

0|..........|

1|..........|

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

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

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

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

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

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

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

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

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

~Object:

x = 1

y = 1

hp = 0

crown = red

id = 1

~Object.

input coordinates of target (separated by space) [q to exit]: 0 2

Selected target is red

input a damage value: 100

0123456789

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

0|..........|

1|..........|

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

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

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

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

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

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

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

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

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

~Object:

x = 0

y = 2

hp = 0

crown = red

id = 3

~Object.

input coordinates of target (separated by space) [q to exit]: 2 2

Selected target is red

input a damage value: 100

0123456789

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

0|..........|

1|..........|

2|..........|

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

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

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

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

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

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

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

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

~Object:

x = 2

y = 2

hp = 0

crown = red

id = 2

~Object.

~Crown:

color = red

~Crown.

**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.

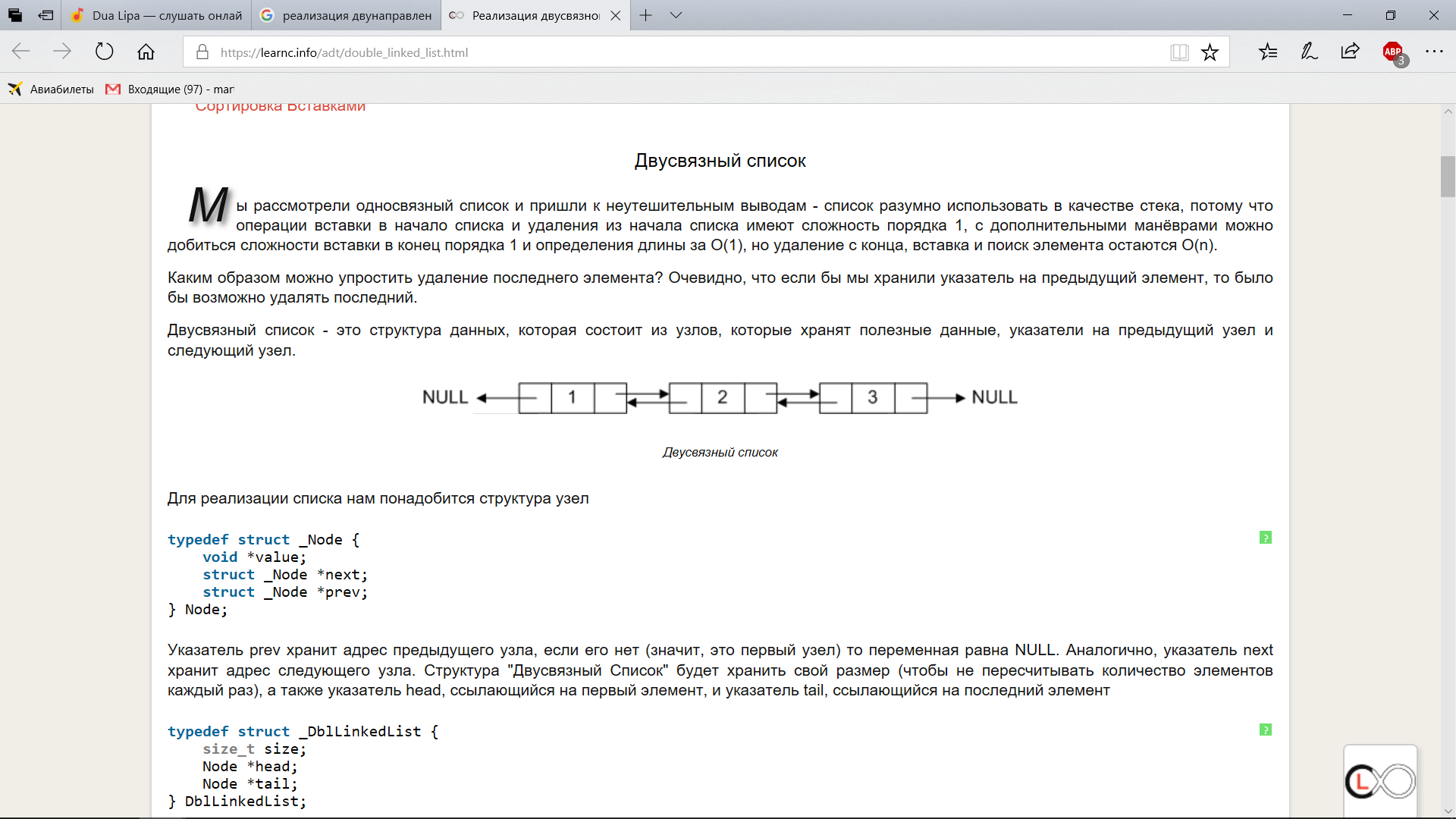
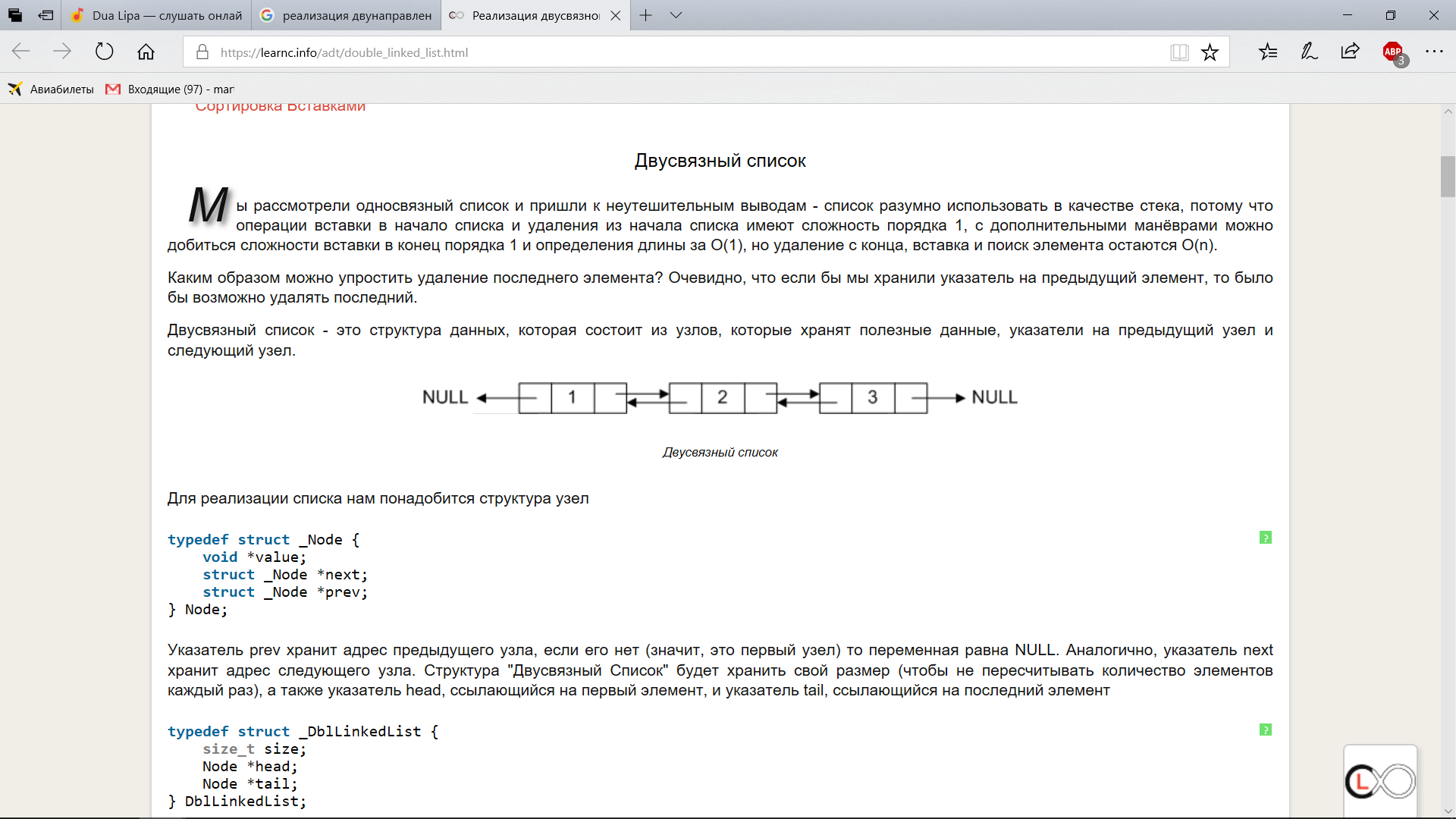


Рисунок 1. Двусвязный список.

**Спецификация программы.**

* Для класса Object определены следующие функции:

1. Говорящие конструктор Object(int x, int y, int hp, const shared\_ptr<Crown> crown) : x(x), y(y), hp(hp), crown(crown), id(count) и деструктор –  и ~Object() соответственно.
2. Функция, инициализирующая поля объекта из файла - Object(ifstream& fl, const shared\_ptr<Crown>& clr) : id(count++), crown(clr).
3. Перегруженный оператор « - ostream& operator << (ostream& str, const Object& obj).
4. Перегруженный оператор » - istream& operator >> (istream &str, Object &obj).
5. Функция, определяющая, есть ли объект на указанной позиции – bool Is\_Located(int x1, int y1).
6. Функция, принимающая урон – int Damage(int damage).
7. Функция, определяющая, убит ли объект – bool Is\_Dead().

* Для класса Field определены следующие функции:

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

* Для класса Crown определены следующие функции:

1. Говорящие конструктор и деструктор Crown() и ~Crown() соотвественно.

2. Функция, задающая цвет короны - void Set\_Color(bool flag).

3. Функция, определяющая цвет короны - string Get\_Color(bool flag).

4. Конструктор, определяющий цвет короны - Crown(bool flag).

* Для класса Iterator определены следующие функции:

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 ++() - оператор-инкремент.
5. Iterator operator --() - 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 функции:

1. ~List() – деструктор.
2. void Print() – выводит на экран список.
3. void Supl\_End(ifstream& fl, const shared\_ptr<Crown>& color) – добавляет элемент в конец списка.
4. void Delete(const T& info) — удаляет элемент.
5. void Supl\_Begin(T info) – добавляет элемент в начало списка.
6. void Supl\_Pos(T info, int pos) – добавляет элемент по позиции.
7. void Erase\_End(T info) – удаляет элемент из конца списка.
8. void Erase\_Begin(T info) – удаляет элемент из начала списка.
9. void Erase\_Pos(T info, int pos) – удаляет элемент по позиции.

**Выводы.**

В ходе данной лабораторной работы были получены навыки работы с умными указателями, 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.