Внедренные объекты.

Объектно-ориентированный однородный двухсвязный список.

Цель - с помощью классов реализовать список объектов типа Circle (круг).
Последовательность действий:

1. Создайте класс Point

Подумайте: какие методы и переменные должны быть определены в этом классе.

2. Создайте класс Circle

Этот класс должен содержать:
- внедренный объект типа Point – центр круга
- радиус

При конструировании объектов класса Circle предусмотрите эффективную инициализацию внедренных объектов.

3. Разработаем двухсвязный список, элементы которого будут являться копиями наших объектов

Замечание: так как в списке будут фигурировать копии объектов – нам все равно, каким образом были созданы объекты-оригиналы (глобальные, локальные, динамические, автоматические неименованные) => манипулируя списком, мы несем ответственность только за уничтожение копий, а оригиналы создаются и уничтожаются независимо.
Список реализуется двумя классами:

3.1. Вспомогательный класс для хранения данных и связей между объектами.

Замечание: вспомогательный класс можно объявить независимо или «встроить» прямо в объявлении класса, реализующего список. Независимо от реализации удобно, чтобы все понятия вспомогательного класса были доступны методам класса List => подумайте как можно это обеспечить. Альтернатива – реализация public методов для доступа к защищенным данным.

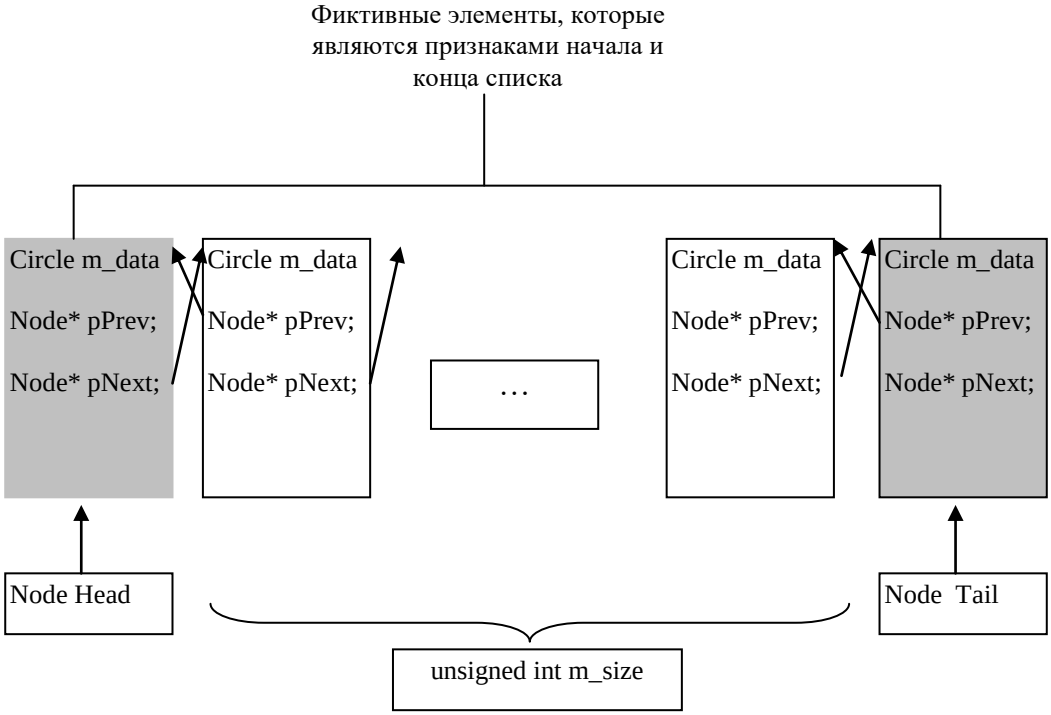
Подсказка:

```
class Node
{
//Данные:
     Node* pPrev;        //указатель на предыдущий элемент
     Node* pNext;        //указатель на следующий элемент
     Circle m_Data;      //Замечание: здесь может быть встроенный
                         объект или указатель на свою (динамическую)
                         копию данных

//Методы:
     Node();//скорее всего понадобится для создания вспомогательных
                         оберток – «стражей» (Head, Tail)
     Node(… ,const Circle *);     //конструктор, посредством которого
                         создаваемый Node «подключается» в список.
     ~Node();//деструктор должен «исключать» уничтожаемый Node из
                         списка
};

//Реализация методов:
//Конструктор
Node::Node(…, const Circle * pc)…//передать параметр pc конструктору
                         встроенного объекта m_Data
```

```
{
     …
}
//Деструктор
Node::~Node()
{
     //Изъяли текущий элемент из списка
     …
}
```

3.2. Основной класс, реализующий список

Теперь вспомогательный класс Node можно использовать при создании связного списка из объектов типа Circle - создадим еще один класс для реализации списка – List.



Подсказка:

```
class List
{
//встроенное объявление класса Node

//данные
     Node Head;     // фиктивный элемент, который является признаком
                       начала списка
     Node Tail;     // фиктивный элемент, который является признаком
                       конца списка
     size_t m_size;//количество элементов
public:
     List() { //сформировать Head, Tail и m_size }
     …
};
```

4. Реализация методов

Реализуйте самостоятельно методы класса List:

1) метод, который добавляет элемент в начало списка
2) метод, который добавляет элемент в конец списка

3) удаление из списка первого элемента, данное которого совпадает со значением параметра. Если элемент найден и удален, метод возвращает true, если элемента с таким значением в списке не было – false
4) удаление из списка всех элементов, данное которых совпадает со значением параметра. Метод возвращает количество удаленных элементов.
5) Сделать список пустым
6) добавьте в класс List остальные, **необходимые** на Ваш взгляд методы.
7)

5. Сортировка для созданного списка и вывод результатов в файл

Напишите для класса List

1. сортировку по возрастанию площади объекта.
2. Вывод текущего состояния списка на консоль

```
cout<<l;
```

3. Вывод текущего состояния списка в файл (в форматированном виде)

```
#include <fstream>
    ofstream fout(ar);
    fout<<l;
    fout.close();
```

4. Чтение из файла

Подсказка:
Для файловых операций воспользуйтесь объектами файлового ввода/вывода (классы ifstream и ofstream).
Пример вывода в файл:

```
{
     cout<<"Enter Output File Name  - " ;
     char ar[80];
     cin>>ar;
     ofstream fout(ar);
     fout<<100<<endl;
     fout.close();
}
```