МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение
высшего профессионального образования
"Донской государственный технический университет"
ДГТУ

Кафедра "Программное обеспечение вычислительной техники и автоматизированных систем"

Динамические структуры данных

Методические указания к лабораторной работе по дисциплине "Объектно-ориентированное программирование"

Ростов-на-Дону
20 г.

Составитель: к.ф.-м.н., доц. Габрельян Б.В.

УДК 512.3

Динамические структуры данных на примере двусвязного списка: методические указания – Ростов н/Д: Издательский центр ДГТУ, 20 . – с.

В методической разработке рассматриваются вопросы реализации в виде классов динамической структуры данных – двусвязного списка. Даны задания по выполнению лабораторной работы. Методические указания предназначены для студентов направления 231000 "Программная инженерия".

I. Объектно-ориентированная реализация динамических структур данных.

В объектно-ориентированной парадигме каждое важное понятие решаемой задачи в общем случае должно быть представлено соответствующим классом. Например, если в программе нужно реализовать двунаправленный список, то этому понятию будет соответствовать класс с подходящим именем, скажем List. Но сам список в своей реализации использует такое важное понятие как узел списка, поэтому при реализации списка удобно создать еще один класс с именем, например, ListNode (узел списка). Этот класс нужен разработчику класса List, но нужен ли он программисту не создающему, а только использующему List? При использовании списка мы можем думать о тех данных, которые мы помещаем или извлекаем из списка, но совсем не интересуемся тем, как организовано хранение этих данные в классе List. Если это так, разработчик может спрятать от пользователя реализацию класса ListNode внутри реализации класса List, сделав последний вложенным закрытым (или защищенным) классом.

II. Представление двунаправленного списка в программе на C++.

Разделяя описание и реализацию, при создании двунаправленного списка его описание помещают в файл заголовков, а реализацию в .cpp-файл, который компилируется и может предоставляться пользователю как часть статической или динамической библиотеки. Описание классов, реализующих двунаправленный список, может быть таким:

Файл list.h

```
class List {
    // закрытый вложенный класс
    class ListNode {
    public:
        int key; // уникальное для каждого узла значение
        char *data; // данные, хранящиеся в узле
```

```
        ListNode *prev, *next; /* указатели на предыдущий и следующий узлы */
    };

    ListNode *first; // указатель на первый узел списка
public:
    List() : first(0) {}
    ~List() { del(); }
    void addData(int key, char *data); // добавить узел
    void removeData(int key); // удалить узел с указанным ключом
    char *findData(int key); // вернуть данные по ключу
    void show(); // отобразить список в консольном окне
private:
    ListNode *findNode(int key); // поиск узла по ключу
    void del(); // удалить все узлы из списка
};
```

III. Двоичное дерево поиска.

Двоичное дерево поиска – это дерево, каждый узел которого имеет не более двух дочерних узлов, каждый узел имеет уникальный ключ, поддерживающий сравнение узлов и узлы упорядочены по ключам. Упорядоченность по ключам в данном случае означает, что для каждого узла выполняется условие – все узлы его левого поддерева имеют значения ключей меньшие, а правого поддерева большие, чем у данного узла. Основными операциями являются операции добавления, удаления и поиска узла.

IV. Задания.

1. Создайте класс List, представляющий понятие "двунаправленный список".

2. Создайте класс, реализующий простое текстовое меню. Используйте его для тестирования созданного в предыдущем задании класса List.

3. Создайте класс Stack, представляющий понятие "стек" (реализующий работу с элементами по дисциплине LIFO – Last In First Out) и поддерживающий простой интерфейс, состоящий из методов push (поместить новый элемент на верхушку стека), pop (вытолкнуть верхний элемент из стека), top (вернуть верхний элемент, не удаляя его из стека) и isEmpty (возвращает булеву величину, true, если стек пуст, false – в противном случае).

4. Измените класс меню из задания 2 для тестирования класса Stack.

5. Создайте класс Queue представляющий понятие "очередь" (реализующий работу с элементами по дисциплине FIFO – First In First Out).

6. Измените класс меню из задания 4 для тестирования класса Queue.

7. Реализуйте класс, представляющий понятие "двоичное дерево поиска". Протестируйте его работу.

V. Контрольные вопросы.

1. Какую дисциплину добавления/удаления элементов реализует очередь?
2. Какую дисциплину добавления/удаления элементов реализует стек?
3. Для чего можно использовать вложенный класс при реализации динамической структуры данных, такой как список?
4. Как организовать класс меню, используя указатели на статические методы класса?
5. Что такое двоичное дерево поиска?
6. Как добавить новый элемент в двоичное дерево поиска?
7. Как найти нужный элемент в двоичном дереве поиска?

Литература

1. Стэнли Б. Липпман. "Язык программирования C++. Вводный курс". – М.: Вильямс. – 2007, 896 с.

2. Г. Шилдт. "Самоучитель C++". – СПб.: БХВ-Петербург. – 2001, 688 с.

3. Стивен Пратта. "Язык программирования C++. Лекции и упражнения", 6-е изд. – М.: Вильямс. – 2012, 1248 с.