Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Объектно-ориентированное программирование»

Студент: Севастьянов В.С.
Преподаватель: Поповкин А. В.
Группа: 08-207

Вариант: 16
Дата:
Оценка:
Подпись:

Москва, 2017

Лабораторная работа №7

1 Цель работы

- Создание сложных динамических структур данных.
- Закрепление принципа OCP.

2 Задача

Необходимо реализовать динамическую структуру данных – "Хранилище объектов"и алгоритм работы с ней. "Хранилище объектов"представляет собой контейнер бинарное дерево. Каждым элементом контейнера является динамическая структура список. Таким образом, у нас получается контейнер в контейнере. Элементов второго контейнера является объект-фигура, определенная вариантом задания.
При этом должно выполняться правило, что количество объектов в контейнере второго уровня не больше 5. Т.е. если нужно хранить больше 5 объектов, то создается еще один контейнер второго уровня.
Объекты в контейнерах второго уровня должны быть отсортированы по возрастанию площади объекта. При удалении объектов должно выполянться правило, что контейнер второго уровня не должен быть пустым. Т.е. если он становится пустым, то он должен удалиться.

Фигуры: Прямоугольник, ромб, трапеция
Контейнер 1: N-дерево **Контейнер 2:** Массив

3 Описание

Принцип открытости/закрытости (OCP) – принцип ООП, устанавливающий следующее положение: "программные сущности (классы, модули, функции и т.п.) должны быть открыты для расширения, но закрыты для изменения".
Контейнер в программировании – структура (АТД), позволяющая инкапсулировать в себе объекты любого типа. Объектами (переменными) контейнеров являются коллекции, которые уже могут содержать в себе объекты определенного типа.
Например, в языке C++, std::list (шаблонный класс) является контейнером, а объект его класса-конкретизации, как например, std::list<int> mylist является коллекцией.
Среди "широких масс"программистов наиболее известны контейнеры, построенные на основе шаблонов, однако, существуют и реализации в виде библиотек (наиболее широко известна библиотека GLib). Кроме того, применяются и узкоспециализированные решения. Примерами контейнеров в C++ являются контейнеры из стандартной библиотеки (STL) – map, vector и т.д. В контейнерах часто встречаются реализации алгоритмов для них. В ряде языков программирования (особенно в скриптовых типа Perl или PHP) контейнеры и работа с ними встроена в язык.
Стек – тип или структура данных в виде набора элементов, которые расположены по принципу LIFO, т.е. "последний пришел, первый вышел". Доступ к элементам осуществляет через обращение к головному элементу (тот, который был добавлен последним).

4 Исходный код

```
1  template <class Container, class T>
2  class Vector
3  {
4  public:
5    Vector();
6
7    void pushBack(std::shared_ptr<Container> value);
8    void print();
9
10   void insertSubitem(std::shared_ptr<T> item, size_t key, size_t parentKey);
11   void removeSubitem(IRemoveCriteria<T> * criteria);
12
13
14   TIterator<VectorItem<Container>, Container> begin();
15   TIterator<VectorItem<Container>, Container> end();
16
17   bool empty();
18
```

```
19    std::shared_ptr<VectorItem<Container>> get(size_t n);
20
21    std::shared_ptr<Container> remove(size_t pos);
22
23    ~Vector();
24  private:
25    void increaseCapacity(size_t n);
26    std::shared_ptr<VectorItem<Container>> *data;
27    size_t capacity;
28    size_t used;
29  };
30
31  template <class T>
32  class VectorItem
33  {
34  public:
35    VectorItem(std::shared_ptr<T> value);
36
37    std::shared_ptr<T> getValue();
38    std::shared_ptr<VectorItem<T>> setNext(std::shared_ptr<VectorItem<T>> next);
39    std::shared_ptr<VectorItem<T>> getNext();
40    void print();
41    ~VectorItem();
42  private:
43    std::shared_ptr<T> _value;
44    std::shared_ptr<VectorItem<T>> _next;
45  };
```

5 Выводы

В этой работе был реализован контейнер второго уровня - динамический массив. В его элементах хранятся N-деревья, с количеством вершин не более 5. Так же было реализовано два вида удаления из дерева - удаление всех фигур одного типа и удаление всех фигур, совпадающих с заданной. В результате работы продемонстрирована возможность хранения и использования структур в структурах.