Лабораторная работа №7

Задача: Необходимо реализовать динамическую структуру данных – "Хранилище объектов"и алгоритм работы с ней. "Хранилище объектов"представляет собой контейнер стек. 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: трапеция, ромб, пятиугольник.
Контейнер 1-ого уровня: связный список.
Контейнер 2-ого уровня: стек.

1 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).

2 Исходный код

Описание классов фигур и класса-контейнера списка остается неизменным.

TStack.hpp	
TStack();	Конструктор класса
void Push(const O&);	Добавление элемента в стек
void Print();	Печать стека
void RemoveByType(const int&);	Удаление из стека элементов одного типа
void RemoveLesser(const double&);	Удаление из стека элементов, площадь которых меньше, чем заданная
virtual ~TStack();	Деконструктор класса

```
1
2 template <typename Q, typename O> class TStack
3 {
```

```
4  private:
5      class Node {
6      public:
7          Q data;
8          std::shared_ptr<Node> next;
9          Node();
10         Node(const Q&);
11         int itemsInNode;
12     };
13
14     std::shared_ptr<Node> head;
15     int count;
16 public:
17     TStack();
18
19     void Push(const Q&);
20     void Print();
21     void RemoveByType(const int&);
22     void RemoveLesser(const double&);
23
24     virtual ~TStack();
25 };
26
27 template <typename T> class TList
28 {
29 private:
30     class TNode {
31     public:
32         TNode();
33         TNode(const std::shared_ptr<T>&);
34         auto GetNext() const;
35         auto GetItem() const;
36         std::shared_ptr<T> item;
37         std::shared_ptr<TNode> next;
38
39         void* operator new(size_t);
40         void operator delete(void*);
41         static TAllocator nodeAllocator;
42     };
43
44     template <typename N, typename M>
45     class TIterator {
46     private:
47         N nodePtr;
48     public:
49         TIterator(const N&);
50         std::shared_ptr<M> operator* ();
51         std::shared_ptr<M> operator-> ();
52         void operator ++ ();
```

```
53          bool operator == (const TIterator&);
54          bool operator != (const TIterator&);
55      };
56
57      int length;
58
59      std::shared_ptr<TNode> head;
60      auto psort(std::shared_ptr<TNode>&);
61      auto pparsort(std::shared_ptr<TNode>& head);
62      auto partition(std::shared_ptr<TNode>&);
63
64  public:
65      TList();
66      virtual ~TList();
67      bool PushFront(const std::shared_ptr<T>&);
68      bool Push(const std::shared_ptr<T>&, const int);
69      bool PopFront();
70      bool Pop(const int);
71      bool IsEmpty() const;
72      int GetLength() const;
73      auto& getHead();
74      auto&& getTail();
75      void sort();
76      void parSort();
77
78      TIterator<std::shared_ptr<TNode>, T> begin() {return TIterator<std::shared_ptr<
            TNode>, T>(head->next);};
79      TIterator<std::shared_ptr<TNode>, T> end() {return TIterator<std::shared_ptr<TNode
            >, T>(nullptr);};
80
81      template <typename A> friend std::ostream& operator<< (std::ostream&, TList<A>&);
82  };
```

3 Консоль

```
karma@karma:~/mai_study/OOP/lab7$ ./run
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
1
Enter bigger base: 2
```

```
Enter smaller base: 1
Enter left side: 1
Enter right side: 1
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
2
Enter side: 3
Enter smaller angle: 10
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
3
Enter side: 3
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
3
Enter side: 4
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
```

```
0) Exit
3
Enter side: 5
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
5
Side = 3,smaller_angle = 10,square = 1.56283,type: rhomb
Smaller base = 1,bigger base = 2,left side = 1,right side = 1,square = 1.86603,type:
trapeze
Sides =  3,square = 15.4843,type: pentagon
Sides =  4,square = 27.5276,type: pentagon
Sides =  5,square = 43.0119,type: pentagon

Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
1
Enter bigger base: 4
Enter smaller base: 3
Enter left side: 3
Enter right side: 3
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
2
Enter side: 4
```

```
Enter smaller angle: 90
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
3
Enter side: 8
Item was added
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
5
Smaller base = 3,bigger base = 4,left side = 3,right side = 3,square = 5.95804,type:
trapeze
Side = 4,smaller_angle = 90,square = 16,type: rhomb
Sides =  8,square = 110.111,type: pentagon

Side = 3,smaller_angle = 10,square = 1.56283,type: rhomb
Smaller base = 1,bigger base = 2,left side = 1,right side = 1,square = 1.86603,type:
trapeze
Sides =  3,square = 15.4843,type: pentagon
Sides =  4,square = 27.5276,type: pentagon
Sides =  5,square = 43.0119,type: pentagon

Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
4
Enter criteria
```

```
1) by type
2) lesser than square
1
1) trapeze
2) rhomb
3) pentagon
Enter type
1
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
5
Side = 4,smaller_angle = 90,square = 16,type: rhomb
Sides =  8,square = 110.111,type: pentagon

Side = 3,smaller_angle = 10,square = 1.56283,type: rhomb
Sides =  3,square = 15.4843,type: pentagon
Sides =  4,square = 27.5276,type: pentagon
Sides =  5,square = 43.0119,type: pentagon

Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
4
Enter criteria
1) by type
2) lesser than square
2
Enter square
20
Choose an operation:
1) Add trapeze
2) Add rhomb
```

```
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
5
Sides =  8,square = 110.111,type: pentagon

Sides =  4,square = 27.5276,type: pentagon
Sides =  5,square = 43.0119,type: pentagon

Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete by criteria
5) Print
0) Exit
0
```