Лабораторная работа №5

Задача: Используя структуры данных, разработанные для предыдущей лабораторной работы (ЛР №4) спроектировать и разработать Итератор для динамической структуры данных.

Итератор должен быть разработан в виде шаблона и должен уметь работать со всеми типами фигур, согласно варианту задания.

Итератор должен позволять использовать структуру данных в операторах типа for. Например: for(auto i : stack) std::cout « *i « std::endl;

Фигуры: трапеция, ромб, пятиугольник.
Контейнер: связный список.

1 Описание

Для доступа к элементам некоторого множества элементов используют специальные объекты, называемые итераторами. В контейнерных типах stl они доступны через методы класса (например, begin() в шаблоне класса vector). Функциональные возможности указателей и итераторов близки, так что обычный указатель тоже может использоваться как итератор.

Категории итераторов:

- Итератор ввода (input iterator) – используется потоками ввода.
- Итератор вывода (output iterator) – используется потоками вывода.
- Однонаправленный итератор (forward iterator) – для прохода по элементам в одном направлении.
- Двунаправленный итератор (bidirectional iterator) – способен пройти по элементам в любом направлении. Такие итераторы реализованы в некоторых контейнерных типах stl (list, set, multiset, map, multimap).
- Итераторы произвольного доступа (random access) – через них можно иметь доступ к любому элементу. Такие итераторы реализованы в некоторых контейнерных типах stl (vector, deque, string, array).

2 Исходный код

Описание классов фигур и класса-контейнера остается неизменным.

```
1
2  template <class N, class T>
3  class TIterator
4  {
5  public:
6      TIterator(std::shared_ptr<N> n) {
7          cur = n;
8      }
9
10     std::shared_ptr<T> operator* () {
11         return cur->GetFigure();
12     }
13
14     std::shared_ptr<T> operator-> () {
15         return cur->GetFigure();
16     }
```

```
17
18      void operator++() {
19          cur = cur->GetNext();
20      }
21
22      TIterator operator++ (int) {
23          TIterator cur(*this);
24          ++(*this);
25          return cur;
26      }
27
28      void operator--() {
29          cur = cur->GetPrev();
30      }
31
32      TIterator operator-- (int) {
33          TIterator cur(*this);
34          --(*this);
35          return cur;
36      }
37
38      bool operator== (const TIterator &i) {
39          return (cur == i.cur);
40      }
41
42      bool operator!= (const TIterator &i) {
43          return (cur != i.cur);
44      }
45
46  private:
47      std::shared_ptr<N> cur;
48  };
```

3 Консоль

```
karma@karma:~/mai_study/OOP/lab5$ ./run
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
2
```

```
Enter side: 10
Enter smaller angle: 10
Enter index = 0
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
2
Enter side: 9
Enter smaller angle: 20
Enter index = 0
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
1
Enter bigger base: 9
Enter smaller base: 8
Enter left side: 7
Enter right side: 6
Enter index = 1
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
3
Enter side: 5
Enter index = 0
Choose an operation:
```

```
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
5
idx: 0   Sides =  5,type: pentagon

idx: 1   Side = 9,smaller_angle = 20,type: rhomb

idx: 2   Side = 10,smaller_angle = 10,type: rhomb

idx: 3   Smaller base = 8,bigger base = 9,left side = 7,right side = 6,type:
trapeze


Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
4
Enter index = 2
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
6
Sides =  5,type: pentagon
Side = 9,smaller_angle = 20,type: rhomb
Smaller base = 8,bigger base = 9,left side = 7,right side = 6,type: trapeze
Choose an operation:
```

```
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
6) Print list with iterator
0) Exit
0
```