Лабораторная работа №4

Целью лабораторной работы является:

- Знакомство с шаблонами классов.
- Построение шаблонов динамических структур данных.

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: октагон, квадрат, треугольник.
Контейнер: связный список.

1 Описание

Шаблоны (template) предназначены для кодирования обобщенных алгоритмов, без привязки к некоторым параметрам (например, типам данных, размерам буферов, значениям по умолчанию). В C++ возможно создание шаблонов функций и классов. Шаблоны позволяют создавать параметризованные классы и функции. Параметром может быть любой типа или значение одного из допустимых типов (целое число, перечисляемый тип, указатель на любой объект с глобально доступным именем, ссылка). Шаблоны особенно полезны при работе с коллекциями и умными указателями.

2 Исходный код

TListItem.cpp	
TListItem(const std::sharedptr<T>&obj);	Конструктор класса
std::sharedptr<T> GetFigure() const;	Получение фигуры из узла
std::sharedptr<TListItem<T» GetNext();	Получение ссылки на следующий узел
void SetNext(std::sharedptr<TListItem<T»item);	Установка ссылки на следующий узел
friend std::ostream& operator«(std::ostream &os, const TListItem<A> &obj);	Переопределенный оператор вывода в поток std::ostream
virtual ~TListItem();	Деконструктор класса
TList.cpp	
TList();	Конструктор класса
void Push(std::sharedptr<T> &obj);	Добавление фигуры в список
std::sharedptr<T> Pop();	Получение фигуры из списка
const bool IsEmpty() const;	Проверка, пуст ли список
uint32t GetLength();	Получение длины списка
friend std::ostream& operator«(std::ostream &os, const TList<A> &list);	Переопределенный оператор вывода в поток std::ostream
virtual ~TList();	Деконструктор класса

```
1
2 template <class T>
3 class TList
4 {
```

```
5  public:
6      TList();
7      void Push(std::shared_ptr<T> &obj);
8      const bool IsEmpty() const;
9      uint32_t GetLength();
10     std::shared_ptr<T> Pop();
11     template <class A> friend std::ostream& operator<<(std::ostream &os, const TList<A>
           &list);
12     virtual ~TList();
13
14 private:
15     uint32_t length;
16     std::shared_ptr<TListItem<T>> head;
17
18     void PushFirst(std::shared_ptr<T> &obj);
19     void PushLast(std::shared_ptr<T> &obj);
20     void PushAtIndex(std::shared_ptr<T> &obj, int32_t ind);
21     std::shared_ptr<T> PopFirst();
22     std::shared_ptr<T> PopLast();
23     std::shared_ptr<T> PopAtIndex(int32_t ind);
24 };
25
26 template <class T>
27 class TListItem
28 {
29 public:
30     TListItem(const std::shared_ptr<T> &obj);
31
32     std::shared_ptr<T> GetFigure() const;
33     std::shared_ptr<TListItem<T>> GetNext();
34     void SetNext(std::shared_ptr<TListItem<T>> item);
35     template <class A> friend std::ostream& operator<<(std::ostream &os, const
           TListItem<A> &obj);
36
37     virtual ~TListItem(){};
38
39 private:
40     std::shared_ptr<T> item;
41     std::shared_ptr<TListItem<T>> next;
42 };
```

3 Консоль

```
denis@ubuntu:~/Desktop/OOP/lab3$ ./run
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
```

```
4) Delete figure from list
5) Print list
0) Exit
1
Enter side A: 1
Enter side B: 2
Enter side C: 3
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
2
Enter side: 5
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
3
Enter side: 10
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
5
idx: 0   Side A = 1,side B = 2,side C = 3,type: Triangle

idx: 1   Side = 5,type: Foursquare

idx: 2   Side = 10,type: Octagon


Choose an operation:
```

```
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
4
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
5
idx: 0   Side A = 1,side B = 2,side C = 3,type: Triangle

idx: 1   Side = 5,type: Foursquare


Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
3
Enter side: 123
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
5
idx: 0   Side A = 1,side B = 2,side C = 3,type: Triangle

idx: 1   Side = 5,type: Foursquare
```

```
idx: 2   Side = 123,type: Octagon


Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
4
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
4
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
4
Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
4) Delete figure from list
5) Print list
0) Exit
5
The list is empty.

Choose an operation:
1) Add triangle
2) Add foursquare
3) Add octagon
```

```
4) Delete figure from list
5) Print list
0) Exit
0
```

4 Выводы

В данной лабораторной работе необходимо было реализовывать шаблоны классов. Был построен шаблон динамической структуры данных.