Лабораторная работа №4

Задача: Необходимо спроектировать и запрограммировать на языке C++ шаблон класса-контейнера первого уровня, содержащий все три фигуры, согласно варианту задания

Классы должны удовлетворять следующим правилам:

- Требования к классу фигуры аналогичны требованиям из лабораторной работы 1.
- Шаблон класса-контейнера должен содержать объекты, используя std::shared ptr<...>.
- Шаблон класса-контейнера должен иметь метод по добалению фигуры в контейнер.
- Шаблон класса-контейнера должен иметь методы по получению фигуры из контейнера.
- Шаблон класса-контейнера должен иметь метод по удалению фигуры из контейнера.
- Шаблон класса-контейнера должен иметь перегруженный оператор по выводу контейнера в поток ostream.
- Шаблон класса-контейнера должен иметь деструктор, удаляющий все элементы контейнера.
- Классы должны быть расположены в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Фигуры: трапеция, ромб, пятиугольник
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 используются в случаях дублирования одного и того же кода для нескольких типов. Например, можно использовать шаблоны функций для создания набора функций, которые применяют один и тот же алгоритм к различным типам данных. Кроме того, шаблоны классов можно использовать для разработки набора типобезопасных классов. Иногда рекомендуется использовать шаблоны вместо макросов C и пустых указателей. Шаблоны особенно полезны при работы 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
std::sharedptr<TListItem<T» GetPrev();	Получение ссылки на предыдущий узел
void SetNext(std::sharedptr<TListItem<T» item);	Установка ссылки на следующий узел
void SetPrev(std::sharedptr<TListItem<T» item);	Установка ссылки на предыдущий 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     void Del();
13     virtual ~TList();
14
15 private:
16     uint32_t length;
17     std::shared_ptr<TListItem<T>> head;
18
19     void PushFirst(std::shared_ptr<T> &obj);
20     void PushLast(std::shared_ptr<T> &obj);
21     void PushAtIndex(std::shared_ptr<T> &obj, int32_t ind);
22     std::shared_ptr<T> PopFirst();
23     std::shared_ptr<T> PopLast();
24     std::shared_ptr<T> PopAtIndex(int32_t ind);
25 };
26
27 template <class T>
28 class TListItem
29 {
30 public:
31     TListItem(const std::shared_ptr<T> &obj);
32
33     std::shared_ptr<T> GetFigure() const;
34     std::shared_ptr<TListItem<T>> GetNext();
35     std::shared_ptr<TListItem<T>> GetPrev();
36     void SetNext(std::shared_ptr<TListItem<T>> item);
37     void SetPrev(std::shared_ptr<TListItem<T>> item);
38     template <class A> friend std::ostream& operator<<(std::ostream &os, const
```

```
        TListItem<A> &obj);
39
40      virtual ~TListItem(){};
41
42  private:
43      std::shared_ptr<T> item;
44      std::shared_ptr<TListItem<T>> next;
45      std::shared_ptr<TListItem<T>> prev;
46  };
```

3 Консоль

```
karma@karma:~/mai_study/OOP/lab4$ ./run
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
1
Enter bigger base: 10
Enter smaller base: 5
Enter left side: 5
Enter right 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
0) Exit
1
Enter bigger base: 15
Enter smaller base: 10
Enter left side: 5
Enter right 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
0) Exit
2
Enter side: 10
Enter smaller angle: 60
Enter index = 1
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
3
Enter side: 3
Enter index = 2
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
5
idx: 0   Smaller base = 10,bigger base = 15,left side = 5,right side = 5,type:
trapeze

idx: 1   Smaller base = 5,bigger base = 10,left side = 5,right side = 5,type:
trapeze

idx: 2   Side = 10,smaller_angle = 60,type: rhomb

idx: 3   Sides =  3,type: pentagon


Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
```

```
4) Delete figure from list
5) Print list
0) Exit
4
Enter index = 0
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
4
Enter index = 1
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
4
Enter index = 1
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
4
Enter index = 0
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
5
The list is empty.
```

```
Choose an operation:
1) Add trapeze
2) Add rhomb
3) Add pentagon
4) Delete figure from list
5) Print list
0) Exit
0
```