

## Лабораторная работа №3

**Задача:** Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий все три фигуры, согласно варианту задания. Классы должны удовлетворять следующим правилам:

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

**Фигуры:** трапеция, ромб, пятиугольник

**Контейнер:** связный список.

# 1 Описание

Умный указатель – класс (обычно шаблонный), имитирующий интерфейс обычного указателя и добавляющий некую новую функциональность, например, проверку границ при доступе или очистку памяти.

Существует 3 вида умных указателей стандартной библиотеки C++:

- `unique ptr` – обеспечивает, чтобы у базового указателя был только один владелец. Может быть передан новому владельцу, но не может быть скопирован или сделан общим. Заменяет `auto ptr`, использовать который не рекомендуется.
- `shared ptr` – умный указатель с подсчитанными ссылками. Используется, когда необходимо присвоить один необработанный указатель нескольким владельцам, например, когда копия указателя возвращается из контейнера, но требуется сохранить оригинал. Необработанный указатель не будет удален до тех пор, пока все владельцы `shared ptr` не выйдут из области или не откажутся от владения.
- `weak ptr` – умный указатель для особых случаев использования с `shared ptr`. `weak ptr` предоставляет доступ к объекту, который принадлежит одному или нескольким экземплярам `shared ptr`, но не участвует в подсчете ссылок. Используется, когда требуется отслеживать объект, но не требуется, чтобы он оставался в активном состоянии.

# 2 Исходный код

TListItem.cpp	
<code>TListItem(const std::sharedptr&lt;Figure&gt; &amp;obj);</code>	Конструктор класса
<code>std::sharedptr&lt;Figure&gt; GetFigure() const;</code>	Получение фигуры из узла
<code>std::sharedptr&lt;TListItem&gt; GetNext();</code>	Получение ссылки на следующий узел
<code>std::sharedptr&lt;TListItem&gt; GetPrev();</code>	Получение ссылки на предыдущий узел
<code>void SetNext(std::sharedptr&lt;TListItem&gt; item);</code>	Установка ссылки на следующий узел
<code>void SetPrev(std::sharedptr&lt;TListItem&gt; item);</code>	Установка ссылки на предыдущий узел
<code>friend std::ostream&amp; operator&lt;&lt;(std::ostream &amp;os, const TListItem &amp;obj);</code>	Переопределенный оператор вывода в поток <code>std::ostream</code>
<code>virtual ~TListItem();</code>	Деконструктор класса

TList.cpp	
TList();	Конструктор класса
void Push(std::shared_ptr<Figure> &obj);	Добавление фигуры в список
std::shared_ptr<Figure> Pop();	Получение фигуры из списка
const bool IsEmpty() const;	Проверка, пуст ли список
uint32_t GetLength();	Получение длины списка
friend std::ostream& operator<<(std::ostream &os, const TList &list);	Переопределенный оператор вывода в поток std::ostream
virtual ~TList();	Деконструктор класса

```

1
2 class TList
3 {
4 public:
5     TList();
6     void Push(std::shared_ptr<Figure> &obj);
7     const bool IsEmpty() const;
8     uint32_t GetLength();
9     std::shared_ptr<Figure> Pop();
10    friend std::ostream& operator<<(std::ostream &os, const TList &list);
11    virtual ~TList();
12
13 private:
14     uint32_t length;
15     std::shared_ptr<TListItem> head;
16
17     void PushFirst(std::shared_ptr<Figure> &obj);
18     void PushLast(std::shared_ptr<Figure> &obj);
19     void PushAtIndex(std::shared_ptr<Figure> &obj, int32_t ind);
20     std::shared_ptr<Figure> PopFirst();
21     std::shared_ptr<Figure> PopLast();
22     std::shared_ptr<Figure> PopAtIndex(int32_t ind);
23 };
24
25 class TListItem
26 {
27 public:
28     TListItem(const std::shared_ptr<Figure> &obj);
29
30     std::shared_ptr<Figure> GetFigure() const;
31     std::shared_ptr<TListItem> GetNext();
32     std::shared_ptr<TListItem> GetPrev();
33     void SetNext(std::shared_ptr<TListItem> item);
34     void SetPrev(std::shared_ptr<TListItem> item);
35     friend std::ostream& operator<<(std::ostream &os, const TListItem &obj);

```

```

36 |
37 |     virtual ~TListItem(){};
38 |
39 | private:
40 |     std::shared_ptr<Figure> item;
41 |     std::shared_ptr<TListItem> next;
42 |     std::shared_ptr<TListItem> prev;
43 | };

```

### 3 КОНСОЛЬ

```
karma@karma:~/mai_study/OOP/lab3$ ./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: 5

Enter smaller base: 2

Enter left side: 2

Enter right side: 2

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

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
- 0) Exit

3

Enter side: 10

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

2

Enter side: 6

Enter smaller angle: 40

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

5

idx: 0    Sides = 5,type: pentagon

idx: 1    Side = 6,smaller\_angle = 40,type: rhomb

idx: 2    Smaller base = 2,bigger base = 5,left side = 2,right side = 2,type:  
trapeze

idx: 3    Sides = 10,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 = 2

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

idx: 0 Side = 6,smaller\_angle = 40,type: rhomb

idx: 1 Sides = 10,type: pentagon

Choose an operation:

- 1) Add trapeze
- 2) Add rhomb
- 3) Add pentagon
- 4) Delete figure from list
- 5) Print list
- 0) Exit

0