Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 3

Студент: Бронников М. А.
Группа: 80-204
Преподаватель: Чернышов Л.Н.
Дата: 21.10.2018
Оценка:

Москва, 2018

Вариант 1:

Контейнер первого уровня массив

Контейнер второго уровня массив

Фигуры: треугольник, прямоугольник, квадрат.

Целью лабораторной работы является:

Закрепление навыков работы с классами.

Знакомство с умными указателями.

Задание

Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий **все три** фигуры класса фигуры, согласно вариантов задания (реализованную в ЛР1).

Классы должны удовлетворять следующим правилам:

- Требования к классу фигуры аналогичны требованиям из лабораторной работы 1.
- Класс-контейнер должен соджержать 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 (опередеяется структурой контейнера).
- Класс-контейнер должен иметь метод по удалению фигуры из контейнера (опередеяется структурой контейнера).
- Класс-контейнер должен иметь перегруженный оператор по выводу контейнера в поток std::ostream (<<).
- Класс-контейнер должен иметь деструктор, удаляющий все элементы контейнера.
- Классы должны быть расположенны в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Нельзя использовать:

- Стандартные контейнеры std.
- Шаблоны (template).
- Объекты «по-значению»

Программа должна позволять:

- Вводить произвольное количество фигур и добавлять их в контейнер.
- Распечатывать содержимое контейнера.
- Удалять фигуры из контейнера.

Модули программы:

Figure.h	Родительский класс фигур
Triangle.h, Triangle.cpp	Описание трегольника
Rectangle.h, Rectangle.cpp	Описание прямоугольника

Quadrate.h, Quadrate.cpp	Описание квадрата
BinTreeItem.h, BinTreeItem.cpp	Шаблон элемента
BinTree.h, BinTree.cpp	Шаблон
main.cpp	Интерфейс

Описание классов:

Figure

virtual double Square()	Вычисление площади
virtual bool IsCorrect()	Проверка на корректность ввода
virtual void Print()	Вывод фигуры на экран
virtual ~Figure()	Деструктор

Triangle

Triangle()	Конструктор по умолчанию
Triangle(double i, double j, double k, double l)	Конструктор с 4 аргументами
Triangle(constTriangle& orig)	Конструктор копирования
Triangle(std::istream& is)	Конструктор потокового ввода
double Square()	Вычисление площади
void Print()	Вывод на экран
ostream& operator<<(std::ostream& os,Triangle& obj)	Перегруженный оператор вывода на экран
istream& operator>>(std::istream& is,Triangle& obj)	Перегруженный оператор ввода сторон
Triangle& operator=(const Triangle& right)	Перегруженный оператор присваивания
virtual ~Triangle()	Декструктор
bool IsCorrect()	Проверка фигуры на корректность ввода
double side_a, side_b, side_c, side_D	Стороны трегольника

Rectangle

Rectangle()	Конструктор по умолчанию
Rectangle(std::istream &is)	Конструктор потокового ввода
Rectangle(double i, double j)	Конструктор с двумя аргументами
Rectangle(const Rectangle& orig)	Конструктор копирования
istream& operator>>(std::istream& is, Rectangle& obj)	Перегруженный оператор ввода сторон
ostream& operator<<(std::ostream& os, Rectangle& obj)	Перегруженный оператор вывода на экран
double Square()	Вычисление площади
bool IsCorrect()	Проверка фигуры на корректность ввода
void Print()	Вывод на экран
virtual ~Rectangle()	Деструктор
double side_a, side_b	Стороны прямоугольника

Quadrate

Quadrate()	Конструктор по умолчанию
Quadrate(std::istream &is)	Конструктор копирования
Quadrate(double i, double j)	Конструктор с 2 аргументами
Quadrate(const Quadrate& orig)	Конструктор копирования
istream& operator>>(std::istream& is, Quadrate& obj)	Перегруженный оператор ввода сторон
ostream& operator<<(std::ostream& os, Quadrate& obj)	Перегруженный оператор вывода на экран
double Square()	Вычисление площади
bool IsCorrect()	Проверка фигуры на корректность ввода
void Print()	Вывод на экран
virtual ~Sqr()	Деструктор

double side_d1, side_d2	Диагонали квадрата

BinTreeItem

BinTreeItem(const shared_ptr<T>& obj)	Конструктор
BinTreeItem(const shared_ptr<BinTreeItem<T>>& orig)	Конструктор копирования
ostream& operator<<(ostream& os, const BinTreeItem<A>& obj)	Перегруженный оператор вывода узла на экран
shared_ptr<BinTreeItem<T>> SetRight(shared_ptr<BinTreeItem<T>> right)	Присвоить узел правому поддерову
shared_ptr<BinTreeItem<T>> SetLeft(shared_ptr<BinTreeItem<T>> left)	Присвоить узел левому поддерову
shared_ptr<BinTreeItem<T>> GetRight()	Взятие ссылки на правое поддерево
shared_ptr<BinTreeItem<T>> GetLeft()	Взятие ссылки на левое поддерево
shared_ptr<BinTreeItem<T>> GetParent()	Взятие ссылки на родителя
shared_ptr<BinTreeItem<T>> SetParent(shared_ptr<BinTreeItem<T>> parent)	Присвоить узел родителю
shared_ptr<T> GetFigure() const	Взятие фигуры из узла
shared_ptr<T> SetFigure(shared_ptr<T> figure)	Присвоить фигуру узлу
virtual ~BinTreeItem()	Деструктор
shared_ptr<T> figure	Поле данных
shared_ptr<BinTreeItem<T>> left, shared_ptr<BinTreeItem<T>> right	Ссылки на левое, правое поддерево
shared_ptr<BinTreeItem<T>> parent	Ссылка на родительский узел

BinTree

BinTree()	Конструктор
shared_ptr<BinTreeItem<T>> Search(shared_ptr<T> figure, shared_ptr<BinTreeItem<T>> leaf)	Поиск элемента в
void Print(shared_ptr<BinTreeItem<T>> leaf)	Вывод на экран
void Insert(shared_ptr<T> figure, shared_ptr<BinTreeItem<T>> leaf)	Вставить элемент

shared_ptr<BinTreeItem<T>> Minimum(shared_ptr<BinTreeItem<T>> leaf)	Поиск минимального элемента
shared_ptr<BinTreeItem<T>> Maximum(shared_ptr<BinTreeItem<T>> leaf)	Поиск максимального элемента
shared_ptr<BinTreeItem<T>> Succesor(shared_ptr<BinTreeItem<T>> leaf)	Поиск преемника
void Delete(shared_ptr<BinTreeItem<T>> leaf)	Удаление элемента
virtual ~BinTree()	Деструктор

1. Решение задачи и правила использования

Я усовершенствовал мой класс-контейнер. Теперь он может хранить в себе все три типа фигур. Это было реализована при помощи shared_ptr.

Конструктор класса:

Triangle ();

Конструктор класса из стандартного потока:

Triangle(std::istream &is);

Конструктор копии класса:

Triangle(const Triangle& orig):

Площадь фигуры:

double Square():

Печать фигуры:

void Print();

Деструктор класса.

~Triangle();

Переопределенные операторы:

Переопределенный оператор вывода <<:

friend std::ostream& operator<<(std::ostream& os, const Triangle& obj);

Переопределенный оператор ввода >> :

friend std::istream& operator>>(std::istream& is, Triangle& obj)

Переопределенный оператор присваивания = :

Triangle& operator=(const Triangle& right);
Переопределенный оператор сравнения ==
friend bool operator==(const Triangle& rigth, const Triangle& left);

Удаление элемента по индексу:
pop(size_t index);
Проверка пустоты элемента
bool empty(size_t index); bool empty();

Переопределенные операторы контейнера
Переопределенный оператор вывода <<:
friend std::ostream& operator<< (std::ostream& os, mass& obj);
Переопределенный оператор ввода >>:
friend std::istream& operator>> (std::istream& is, mass& obj);
Деструктор массива:
~mass();

Правила использования:

Использование в среде линукс Linux.

Все файлы должны быть расположены в одной папке. В интерпретатор команд Bash, находясь в папке с файлом, вводим команду:
g++ -std=c++17 main.cpp Triangle.cpp Hexagon.cpp Octagon.cpp mass.cpp

Чтобы запустить скомпилированную программу:
./a.out
В операционной системе Windows
Выполнить инструкцию на сайте по установке MinGW компилятора
https://metanit.com/cpp/c/1.2.php
Все файлы должны быть расположены в одной папке. В интерпретатор команд cmd.exe, находясь в папке с файлом, вводим команду:

g++ -std=c++17 main.cpp Triangle.cpp Quadrate.cpp Rectangle.cpp massive.cpp

исполнение:

a.exe
Далее в меню выбирать при помощи цифр ход выполнения программы и вводить данные.

4. Тесты:

```
max@max-X550CC:~/oop3$ ./hello
massive created!
massive created!
massive copied!
------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:1
Massive:
Size:10
Elements:
[0]:empty
[1]:empty
[2]:empty
[3]:empty
[4]:empty
[5]:empty
[6]:empty
[7]:empty
[8]:empty
[9]:empty

------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
```

```
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:4
Enter index:5
Enter:
1-If want to add triangle
2-If want to add quadrate
3-If want to add rectangle
Your choice:2
Enter quadrate:
23
------------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:6
Enter index:9
Enter:
1-If want to add triangle
2-If want to add quadrate
3-If want to add rectangle
Your choice:1
Enter triangle:
2 3 4
Triangle created: 2, 3, 4
------------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
```

```
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:7
Enter new size:7
New lenght:7
--------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:4
Enter index:2
Enter:
1-If want to add triangle
2-If want to add quadrate
3-If want to add rectangle
Your choice:2
Enter quadrate:
12
--------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:1
Massive:
Size:7
Elements:
[0]:empty
[1]:empty
```

```
[2]:Quadrate:Size of sides:12

[3]:empty
[4]:empty
[5]:Quadrate:Size of sides:23

[6]:empty

------------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:2
Massive:
Size:0
Elements:
Empty!

------------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:3
Massive:
Size:10
Elements:
[0]:empty
[1]:empty
[2]:empty
[3]:empty
```

```
[4]:empty
[5]:empty
[6]:empty
[7]:empty
[8]:empty
[9]:Trianglea=2, b=3, c=4

------------------------------------------------------------
--------
Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
0-Exit
Enter your choise:0
Exit! Made by Bronnikov(№1 M8O-204)
Triangle deleted
Massive deleted!
Massive deleted!
Quadrate deleted
Quadrate deleted
Massive deleted!
max@max-X550CC:~/oop3$
```

5. Листинг программы:

Figure.cpp

```
#include "Figure.h"
#include <iostream>
#include <cstdlib>

std::ostream& operator<<(std::ostream& os, Figure& obj){
  obj.Print();
  return os;
}
```

Figure.h

```
#include <iostream>
#ifndef FIGURE_H
```

```
#define        FIGURE_H

class Figure {
public:
        virtual double Square() = 0;
        virtual void   Print() = 0;
        friend std::ostream& operator<<(std::ostream& os, Figure& obj);
        virtual ~Figure() {};
};

#endif
```

Quadrate.h

```
#ifndef QUADRATE_H
#define QUADRATE_H
#include <cstdlib>
#include <iostream>
#include "Figure.h"

class Quadrate : public Figure{
public:
        Quadrate();
        Quadrate(std::istream &is);
        Quadrate(size_t i);
        Quadrate(const Quadrate& orig);

        double Square() override;
        void   Print() override;
        friend std::ostream& operator<<(std::ostream& os, const
Quadrate& obj);
        friend std::istream& operator>>(std::istream& is, Quadrate&
obj);

        virtual ~Quadrate();
private:
        size_t side_a;
};

#endif
```

Quadrate.cpp

```
#include "Quadrate.h"
#include <iostream>
#include <cmath>

Quadrate::Quadrate() : Quadrate(0) {
}

Quadrate::Quadrate(size_t i) : side_a(i){
        std::cout << "Quadrate created: " << side_a << std::endl;
}
```

```
Quadrate::Quadrate(std::istream &is) {
  int a;
  is >> a;
  if(a>=0){
        side_a=a;
}
else{
  std::cout << "Quadrate not created!" << '\n';
}
}

Quadrate::Quadrate(const Quadrate& orig) {
        std::cout << "Quadrate copy created" << std::endl;
        side_a = orig.side_a;
}

double Quadrate::Square() {
        return (double)(side_a*side_a);
}

void Quadrate::Print() {
        std::cout << "Quadrate:" << *this << std::endl;
}

Quadrate::~Quadrate() {
        std::cout << "Quadrate deleted" << std::endl;
}

std::ostream& operator<<(std::ostream& os, const Quadrate& obj){
        os << "Size of sides:" << obj.side_a << std::endl;
        return os;
}

std::istream& operator>>(std::istream& is, Quadrate& obj){
        int a;
        is >> a;
        if(a < 0){
                std::cout << "Wrong sizes! Not changed!" << '\n';
        } else{
                obj.side_a=a;
                std::cout << "Quadrate changed!" << '\n';
        }
        return is;
}
```

Rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
#include <cstdlib>
#include <iostream>
#include "Figure.h"

class Rectangle : public Figure{
```

```
public:
	Rectangle();
	Rectangle(std::istream &is);
	Rectangle(size_t i,size_t j);
	Rectangle(const Rectangle& orig);

	double Square() override;
	void   Print() override;
	friend std::ostream& operator<<(std::ostream& os, const
Rectangle& obj);
	friend std::istream& operator>>(std::istream& is, Rectangle&
obj);

	virtual ~Rectangle();
private:
	size_t side_a;
	size_t side_b;
};

#endif
```

Rectangle.cpp

```
#include "Triangle.h"
#include <iostream>
#include <cmath>
#include <stdbool.h>

int max(int a, int b);
int min(int a, int b);

int max(int a, int b){
		return a>b ? a:b;
}
int min(int a, int b){
	return a<b ? a:b;
}


Triangle::Triangle() : Triangle(0, 0, 0) {
}

Triangle::Triangle(size_t ai, size_t bi, size_t ci) {
	if(max(ai, max(bi, ci)) > min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not created!" <<
'\n';
	}	else if((ai>=0) && (bi>=0) && (ci>=0)){
			a=ai;
			b=bi;
			c=ci;
			std::cout << "Triangle created: " << a << ", " <<
b << ", " << c << std::endl;
	} else{
			std::cout << "Wrong sizes! Triangle not created!"
<< '\n';
```

```
		}
}


Triangle::Triangle(std::istream &is) {
	int ai, bi, ci;
	is >> ai;
	is >> bi;
	is >> ci;
	if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not created!" <<
'\n';
	}
	else if(ai>=0&&bi>=0&&ci>0){
		a=ai;
		b=bi;
		c=ci;
		std::cout << "Triangle created: " << a << ", " << b << ",
" << c << std::endl;
	}
	else{
		std::cout << "Wrong sizes! Triangle not created!" <<
'\n';
	}
}


Triangle::Triangle(const Triangle& orig) {
	std::cout << "Triangle copy created" << std::endl;
	a = orig.a;
	b = orig.b;
	c = orig.c;
}

double Triangle::Square(){
	double p = double(a + b + c) / 2.0;
	return sqrt(p * (p - double(a))*(p - double(b))*(p -
double(c)));
}

Triangle& Triangle::operator=(const Triangle& right) {

	if (this == &right){
		return *this;
	}
	a = right.a;
	b = right.b;
	c = right.c;

	return *this;
}

Triangle& Triangle::operator++() {
```

```
	++a;
	++b;
	++c;

	return *this;
}

Triangle operator+(const Triangle& left,const Triangle& right) {

	return Triangle(left.a+right.a,left.b+right.b,left.c+right.c);
}

Triangle::~Triangle() {
	std::cout << "Triangle deleted" << std::endl;
}

std::ostream& operator<<(std::ostream& os, const Triangle& obj) {

	os << "a=" << obj.a << ", b=" << obj.b << ", c=" << obj.c <<
std::endl;
	return os;
}

void Triangle::Print(){
	std::cout << "Triangle" << *this;
	return;
}

bool Triangle::operator==(const Triangle& right){
	return (a==right.a && b==right.b && c==right.c);
}

std::istream& operator>>(std::istream& is, Triangle& obj) {
	int ai, bi, ci;
	is >> ai;
	is >> bi;
	is >> ci;
	if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not changed!" <<
'\n';
	}
	else if(ai>=0&&bi>=0&&ci>0){
		obj.a=ai;
		obj.b=bi;
		obj.c=ci;
		std::cout << "Triangle changed! " << std::endl;
	}
	else{
		std::cout << "Wrong sizes! Triangle not changed!" <<
'\n';
	}

	return is;
}
```

Triangle.h

```
#ifndef TRIANGLE_H
#define	TRIANGLE_H
#include <cstdlib>
#include <iostream>
#include "Figure.h"
#include <stdbool.h>

class Triangle : public Figure{
public:
	Triangle();
	Triangle(std::istream &is);
	Triangle(size_t ai,size_t bi,size_t ci);
	Triangle(const Triangle& orig);

	Triangle& operator++();
	double Square() override;
	void Print() override;
	friend Triangle operator+(const Triangle& left,const Triangle&
right);
	friend std::ostream& operator<<(std::ostream& os, const
Triangle& obj);
	friend std::istream& operator>>(std::istream& is,  Triangle&
obj);

	Triangle& operator=(const Triangle& right);
	bool operator==(const Triangle& right);

	virtual ~Triangle();
private:
	size_t a;
	size_t b;
	size_t c;
};

#endif
```

Triangle.cpp

```
#include "Triangle.h"
#include <iostream>
#include <cmath>
#include <stdbool.h>

int max(int a, int b);
int min(int a, int b);

int max(int a, int b){
		return a>b ? a:b;
}
int min(int a, int b){
	return a<b ? a:b;
}
```

```
Triangle::Triangle() : Triangle(0, 0, 0) {
}

Triangle::Triangle(size_t ai, size_t bi, size_t ci) {
	if(max(ai, max(bi, ci)) > min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not created!" <<
'\n';
	}	else if((ai>=0) && (bi>=0) && (ci>=0)){
			a=ai;
			b=bi;
			c=ci;
			std::cout << "Triangle created: " << a << ", " <<
b << ", " << c << std::endl;
	} else{
			std::cout << "Wrong sizes! Triangle not created!"
<< '\n';
		}
}


Triangle::Triangle(std::istream &is) {
	int ai, bi, ci;
	is >> ai;
	is >> bi;
	is >> ci;
	if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not created!" <<
'\n';
	}
	else if(ai>=0&&bi>=0&&ci>0){
		a=ai;
		b=bi;
		c=ci;
		std::cout << "Triangle created: " << a << ", " << b << ",
" << c << std::endl;
	}
	else{
		std::cout << "Wrong sizes! Triangle not created!" <<
'\n';
	}
}


Triangle::Triangle(const Triangle& orig) {
	std::cout << "Triangle copy created" << std::endl;
	a = orig.a;
	b = orig.b;
	c = orig.c;
}

double Triangle::Square(){
	double p = double(a + b + c) / 2.0;
```

```
	return sqrt(p * (p - double(a))*(p - double(b))*(p -
double(c)));
}

Triangle& Triangle::operator=(const Triangle& right) {

	if (this == &right){
		return *this;
	}
	a = right.a;
	b = right.b;
	c = right.c;

	return *this;
}

Triangle& Triangle::operator++() {

	++a;
	++b;
	++c;

	return *this;
}

Triangle operator+(const Triangle& left,const Triangle& right) {

	return Triangle(left.a+right.a,left.b+right.b,left.c+right.c);
}

Triangle::~Triangle() {
	std::cout << "Triangle deleted" << std::endl;
}

std::ostream& operator<<(std::ostream& os, const Triangle& obj) {

	os << "a=" << obj.a << ", b=" << obj.b << ", c=" << obj.c <<
std::endl;
	return os;
}

void Triangle::Print(){
	std::cout << "Triangle" << *this;
	return;
}

bool Triangle::operator==(const Triangle& right){
	return (a==right.a && b==right.b && c==right.c);
}

std::istream& operator>>(std::istream& is, Triangle& obj) {
	int ai, bi, ci;
	is >> ai;
	is >> bi;
```

```
        is >> ci;
        if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
                std::cout << "Wrong sides! Triangle not changed!" <<
'\n';
        }
        else if(ai>=0&&bi>=0&&ci>0){
                obj.a=ai;
                obj.b=bi;
                obj.c=ci;
                std::cout << "Triangle changed! " << std::endl;
        }
        else{
                std::cout << "Wrong sizes! Triangle not changed!" <<
'\n';
        }

        return is;
}
```

Massive.h

```
#ifndef MASSIVE_H
#define MASSIVE_H
#include "Figure.h"
#include "Triangle.h"
#include <memory>

class TrMassive {
public:

  TrMassive();
  TrMassive(unsigned int l);
  TrMassive(const TrMassive& orig);

  bool Empty();
  friend std::ostream& operator<<(std::ostream& os, const TrMassive&
mass);
  std::shared_ptr<Figure>& operator[](const int index);
  int Lenght();
  void Resize(int l);
  ~TrMassive();

private:
  std::shared_ptr<Figure>* data;
  int len;
};

#endif
```

Massive.cpp

```
#include "Massive.h"
#include "Figure.h"
```

```
#include "Triangle.h"
#include <iostream>
#include <cstdlib>
#include <memory>

TrMassive::TrMassive() : TrMassive(0){}

TrMassive::TrMassive(unsigned int l) {
  data=nullptr;
  len=l;
  if(len>0){
    data = new std::shared_ptr<Figure>[len];
  }
  for (short int i=0; i<len; i++){
    data[i]=nullptr;
  }
  std::cout << "massive created!" << std::endl;
}

TrMassive::TrMassive(const TrMassive& orig) {
  len=orig.len;
  data= new std::shared_ptr<Figure>[len];
  for(short int i=0; i<len; i++){
    data[i]=orig.data[i];
  }
  std::cout << "massive copied!" << '\n';
}

bool TrMassive::Empty(){
  return (len==0);
}

std::shared_ptr<Figure>& TrMassive::operator[](const int index){
  if ((index >= len)||(index < 0)){
    std::cout << "Wrong index! Returning element with index 0!" <<
'\n';
    return data[0];
  }
  return data[index];
}

int TrMassive::Lenght(){
  return len;
}

TrMassive::~TrMassive(){
  delete[] data;
  len=0;
  std::cout << "Massive deleted!" << '\n';
}

void TrMassive::Resize(int l){
  if(l<0){
    std::cout << "Wrong size!" << '\n';
```

```
    return;
  }
  std::shared_ptr<Figure>* data1;
  if(l==0){
    data1 = nullptr;
  } else{
    data1 = new std::shared_ptr<Figure>[l];
  }
  if (l<len){
    for (short int i = 0; i < l; i++) {
      data1[i]=data[i];
    }} else{
      short int i;
      for(i=0; i < len; i++){
        data1[i]=data[i];
      }
      while(i<l){
        data1[i]=nullptr;
        ++i;
      }}
  delete[] data;
  len=l;
  data=data1;
  data1=nullptr;
  return;
}

std::ostream& operator<<(std::ostream& os, const TrMassive& mass){
  std::cout << "Massive:" << '\n';
  std::cout << "Size:" << mass.len << '\n' << "Elements:" << std::endl;
  if(mass.len==0){
    std::cout << "Empty!" << '\n';
  }
  for(short int i=0; i<mass.len; i++){
    std::cout << "[" << i << "]" << ":";
    if(mass.data[i]!=nullptr){
      mass.data[i]->Print();
    }else{
      std::cout << "empty" << '\n';
    }
  }
  return os;
}
```

main.cpp

```
#include "Massive.h"
#include "Triangle.h"
#include "Figure.h"
#include <iostream>
#include <cstdlib>
#include "Rectangle.h"
#include "Quadrate.h"
//Лабораторная работа №3
//Бронников Максим Андреевич М8О-204Б-17
```

```
//Класс контейнер: массив
//Классы фигур: треугольникб прямоугольник, квадрат
/* Необходимо спроектировать и запрограммировать на языке C++
класс-контейнер первого уровня, содержащий все три фигуры класса
фигуры, согласно вариантов задания (реализованную в ЛР1).
Классы должны удовлетворять следующим правилам:
·         Требования к классу фигуры аналогичны требованиям из
лабораторной работы 1.
·         Класс-контейнер должен соджержать 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 (опередеряется структурой контейнера).
·         Класс-контейнер должен иметь метод по удалению фигуры из
контейнера (опередеряется структурой контейнера).
·         Класс-контейнер должен иметь перегруженный оператор по выводу
контейнера в поток std::ostream (<<).
·         Класс-контейнер должен иметь деструктор, удаляющий все
элементы контейнера.
·         Классы должны быть расположенны в раздельных файлах: отдельно
заголовки (.h), отдельно описание методов (.cpp).

Нельзя использовать:
·         Стандартные контейнеры std.
·         Шаблоны (template).
·         Объекты «по-значению»

Программа должна позволять:
·         Вводить произвольное количество фигур и добавлять их в
контейнер.
·         Распечатывать содержимое контейнера.
·         Удалять фигуры из контейнера. */
int main(){
  int i;
  TrMassive mass1(10);
  TrMassive mass2;
  TrMassive mass3(mass1); //тестовые массивы для
  std::shared_ptr<Figure> abc;
  while(1){
   std::cout <<
"--------------------------------------------------------------------------
" << '\n';
   std::cout << "Menu\n1-Print massive №1\n2-Print massive №2\n3-Print
massive №3\n4-Enter figure in №1\n5-Enter figure in №2\n6-Print figure
in №3\n7-Resize №1\n8-Resize №2\n9-Resize №3\n0-Exit\nEnter your
choise:";
   std::cin >> i;
   switch (i) {
    case 1:
      std::cout << mass1 << '\n';
      break;
    case 2:
      std::cout << mass2 << '\n';
```

```
      break;
    case 3:
      std::cout << mass3 << '\n';
      break;
    case 4:
      std::cout << "Enter index:";
      std::cin >> i;
      if(0 < i < mass1.Lenght()){
        short int j;
        std::cout << "Enter:\n1-If want to add triangle\n2-If want to
add quadrate\n3-If want to add rectangle" << '\n';
        std::cout << "Your choice:";
        std::cin >> j;
        if(j==1){
          std::cout << "Enter triangle:" << '\n';
          abc = std::make_shared<Triangle>(std::cin);
          mass1[i] = abc;
        }
        else if(j==2){
          std::cout << "Enter quadrate:" << '\n';
          abc = std::make_shared<Quadrate>(std::cin);
          mass1[i] = abc;
        }
        else if(j==3){
          std::cout << "Enter rectangle:" << '\n';
          abc=std::make_shared<Rectangle>(std::cin);
          mass1[i] = abc;
        } else{
            std::cout << "Wrong choice!" << '\n';
         }
      } else {
        std::cout << "Wrong index!" << '\n';
      }
      break;
    case 5:
      std::cout << "Enter index:";
      std::cin >> i;
      if(0 < i < mass2.Lenght()){
        short int j;
        std::cout << "Enter:\n1-If want to add triangle\n2-If want to
add quadrate\n3-If want to add rectangle" << '\n';
        std::cout << "Your choice:";
        std::cin >> j;
        if(j==1){
          std::cout << "Enter triangle:" << '\n';
          abc = std::make_shared<Triangle>(std::cin);
          mass2[i] = abc;
        }
        else if(j==2){
          std::cout << "Enter quadrate:" << '\n';
          abc = std::make_shared<Quadrate>(std::cin);
          mass2[i] = abc;
        }
        else if(j==3){
```

```
          std::cout << "Enter rectangle:" << '\n';
          abc=std::make_shared<Rectangle>(std::cin);
          mass2[i] = abc;
        } else{
            std::cout << "Wrong choice!" << '\n';
         }
      }else {
        std::cout << "Wrong index!" << '\n';
      }
      break;
    case 6:
      std::cout << "Enter index:";
      std::cin >> i;
      if(0 < i < mass3.Lenght()){
        short int j;
        std::cout << "Enter:\n1-If want to add triangle\n2-If want to
add quadrate\n3-If want to add rectangle" << '\n';
        std::cout << "Your choice:";
        std::cin >> j;
        if(j==1){
          std::cout << "Enter triangle:" << '\n';
          abc = std::make_shared<Triangle>(std::cin);
          mass3[i] = abc;
        }
        else if(j==2){
          std::cout << "Enter quadrate:" << '\n';
          abc = std::make_shared<Quadrate>(std::cin);
          mass3[i] = abc;
        }
        else if(j==3){
          std::cout << "Enter rectangle:" << '\n';
          abc=std::make_shared<Rectangle>(std::cin);
          mass3[i] = abc;
        } else{
            std::cout << "Wrong choice!" << '\n';
         }
      }else {
        std::cout << "Wrong index!" << '\n';
      }
        break;
    case 7:
      std::cout << "Enter new size:";
      std::cin >> i;
      mass1.Resize(i);
      std::cout << "New lenght:" << mass1.Lenght() <<'\n';
      break;
    case 8:
      std::cout << "Enter new size:";
      std::cin >> i;
      mass2.Resize(i);
      std::cout << "New lenght:" << mass2.Lenght() <<'\n';
      break;
    case 9:
      std::cout << "Enter new size:";
```

```
      std::cin >> i;
      mass3.Resize(i);
      std::cout << "New lenght:" << mass3.Lenght() <<'\n';
      break;
    case 0:
      std::cout << "Exit! Made by Bronnikov(№1 M8O-204)" << '\n';
      return 0;
      break;
}}}
```

6. Вывод:

На мой взгляд работа довольно позванавтельная, так как она познакомила меня с контейнерами, закрепила работу с классами, также я узнал про функционал умных указателей, но честно говоря мне сложно представить, где можно использовать данную программу работующую с фигурами, разве что только в образовательных целях.
Спроектировал и запрограммировала на языке C++ класс-контейнер первого уровня, содержащий все три фигуры, согласно варианту задания на основе умных указателей.

СПИСОК ОБРАЗОВАТЕЛЬНЫХ ИСТОЧНИКОВ

- https://ru.stackoverflow.com/questions/605097/%D0%A3%D0%BC%D0%BD%D1%8B%D0%B9-%D1%83%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C-%D0%B8-%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2
- Дейтел Х., Дейтел П. Как программировать на C++ (5-е издание,2008).

- https://rsdn.org/forum/cpp/3979125.all
- https://rsdn.org/forum/cpp/3979125.all