Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Дисциплина: «Объектно-ориентированное программирование»

Отчёт по лабораторным работам

Вариант 6

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №1

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №1

Цели:

- Программирование классов на языке C++
- Управление памятью в языке C++
- Изучение базовых понятий ООП.
- Знакомство с классами в C++.
- Знакомство с перегрузкой операторов.
- Знакомство с дружественными функциями.
- Знакомство с операциями ввода-вывода из стандартных библиотек.

Задание:

Необходимо спроектировать и запрограммировать на языке C++ классы фигур, согласно вариантам заданий. В моём случае – пяти-,шести- и восьмиугольник.

Классы должны удовлетворять следующим правилам:

- Должны иметь общий родительский класс Figure.
- Должны иметь общий виртуальный метод Print, печатающий параметры фигуры и ее тип в стандартный поток вывода cout.
- Должный иметь общий виртуальный метод расчета площади фигуры – Square.
- Должны иметь конструктор, считывающий значения основных параметров фигуры из стандартного потока cin.
- Должны быть расположенны в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Код:

ЛИСТИНГ ФАЙЛА FIGURE.H

```
#ifndef FIGURE_H_INCLUDED
#define    FIGURE_H_INCLUDED


class Figure {
public:
     virtual double Square() = 0;
     virtual void   Print() = 0;
     //virtual void Change() = 0;
     virtual ~Figure() {};
};
```

```
#endif     /* FIGURE_H */
```

ЛИСТИНГ ФАЙЛА MAIN.H

```
//variant №6


#include <cstdlib>
#include "pch.h"
#include "Triangle.h"
#include "Pentagon.h"
#include "Hexagon.h"
#include "Octagon.h"



int main() {


	char c


	std::cout << "Please, type in five points in std coordinate system. Do not forget the order!!" << std::endl;
	Figure *penta_ptr = new Pentagon(std::cin);
	penta_ptr->Print();
	std::cout << penta_ptr->Square() << std::endl;
	delete penta_ptr;


	std::cout << "Please, type in six points in std coordinate system. Do not forget the order!!" << std::endl;
	Figure *hexa_ptr = new Hexagon(std::cin);
	hexa_ptr->Print();
	std::cout << hexa_ptr->Square() << std::endl;
	delete hexa_ptr;
```

```
	std::cout << "Please, type in eight points in std coordinate system. Do not forget the order!!" << std::endl;
	Figure *octa_ptr = new Octagon(std::cin);
	octa_ptr->Print();
	std::cout << octa_ptr->Square() << std::endl;
	delete octa_ptr;

	return 0;
}
```

ЛИСТИНГ ФАЙЛА TRIANGLE.H

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

class Triangle : public Figure {
public:
	Triangle();
	Triangle(std::istream &is);
	Triangle(double i, double j, double k);
	Triangle(const Triangle& orig);

	double Square() override;
	void   Print() override;
	void    Set(double n_a, double n_b, double n_c);
	//void   Change (double a, double b, double c) override;

	virtual ~Triangle();
private:
	double side_a;
```

```
        double side_b;
        double side_c;
};


#endif     /* TRIANGLE_H */
```

ЛИСТИНГ ФАЙЛА TRIANGLE.CPP

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


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


Triangle::Triangle(double i, double j, double k) : side_a(i),
side_b(j), side_c(k) {
        std::cout << "Triangle created: " << side_a << ", " << side_b
<< ", " << side_c << std::endl;
}


Triangle::Triangle(std::istream &is) {
        is >> side_a;
        is >> side_b;
        is >> side_c;
}


Triangle::Triangle(const Triangle& orig) {
        std::cout << "Triangle copy created" << std::endl;
        side_a = orig.side_a;
        side_b = orig.side_b;
        side_c = orig.side_c;
}
```

```
double Triangle::Square() {
	double p = double(side_a + side_b + side_c) / 2.0;
	return sqrt(p * (p - double(side_a))*(p - double(side_b))*(p - double(side_c)));

}

void Triangle::Print() {
	std::cout << "a=" << side_a << ", b=" << side_b << ", c=" << side_c << std::endl;

}

void Triangle::Set(double n_a, double n_b, double n_c) {
	side_a = n_a;
	side_b = n_b;
	side_c = n_c;
}

/*void Triangle::Change(double a, double b, double c){
	side_a = a;
	side_b = b;
	side_c = c;
}*/

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

ЛИСТИНГ ФАЙЛА PENTAGON.H

```
#ifndef PENTAGON_H_INCLUDED
#define	PENTAGON_H_INCLUDED
#include <cstdlib>
```

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


class Pentagon : public Figure {
public:
	Pentagon();
	Pentagon(std::istream &is);
	Pentagon(double i_x, double j_x, double k_x, double l_x, double
m_x, double i_y, double j_y, double k_y, double l_y, double m_y);
	Pentagon(const Pentagon& orig);


	double Square() override;
	void   Print() override;


	virtual ~Pentagon();
private:
	double a_x, a_y;
	double b_x, b_y;
	double c_x, c_y;
	double d_x, d_y;
	double e_x, e_y;
};


#endif     /* Pentagon_H */
```

ЛИСТИНГ ФАЙЛА PENTAGON.CPP

```
#include <cstdlib>
#include <iostream>
#include "Triangle.h"
#include "Pentagon.h"
#include <math.h>
```

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

Pentagon::Pentagon(double i_x, double j_x, double k_x, double l_x,
double m_x, double i_y, double j_y, double k_y, double l_y, double
m_y) :
     a_x(i_x), a_y(i_y), b_x(j_x), b_y(j_y), c_x(k_x), c_y(k_y),
d_x(l_x), d_y(l_y), e_x(m_x), e_y(m_y) {
     std::cout << "Pentagon created:  point A: " << a_x << ", " <<
a_y << std::endl;
     std::cout << "Pentagon created:  point B: " << b_x << ", " <<
b_y << std::endl;
     std::cout << "Pentagon created:  point C: " << c_x << ", " <<
c_y << std::endl;
     std::cout << "Pentagon created:  point D: " << d_x << ", " <<
d_y << std::endl;
     std::cout << "Pentagon created:  point E: " << e_x << ", " <<
e_y << std::endl;
}

Pentagon::Pentagon(std::istream &is) {
     is >> a_x >> a_y;
     is >> b_x >> b_y;
     is >> c_x >> c_y;
     is >> d_x >> d_y;
     is >> e_x >> e_y;
}

Pentagon::Pentagon(const Pentagon& orig) {
     std::cout << "Pentagon copy created" << std::endl;
     a_x = orig.a_x;
     a_y = orig.a_y;
     b_x = orig.b_x;
     b_y = orig.b_y;
```

```
	c_x = orig.c_x;
	c_y = orig.c_y;
	d_x = orig.d_x;
	d_y = orig.d_y;
	e_x = orig.e_x;
	e_y = orig.e_y;
}

double Pentagon::Square() {
	double ab, ac, ae, bc, ad, cd, de;

	ab = sqrt(pow(a_x - b_x, 2) + pow(a_y - b_y, 2));
	ac = sqrt(pow(a_x - c_x, 2) + pow(a_y - c_y, 2));
	ad = sqrt(pow(a_x - d_x, 2) + pow(a_y - d_y, 2));
	ae = sqrt(pow(a_x - e_x, 2) + pow(a_y - e_y, 2));
	bc = sqrt(pow(c_x - b_x, 2) + pow(c_y - b_y, 2));
	cd = sqrt(pow(c_x - d_x, 2) + pow(c_y - d_y, 2));
	de = sqrt(pow(e_x - d_x, 2) + pow(e_y - d_y, 2));

	double res = 0;

	Triangle *T = new Triangle(ab, ac, bc);
	res += T->Square();
	T->Set(ac, cd, ad);
	res += T->Square();
	T->Set(ad, de, ae);
	res += T->Square();
	delete T;
	return res;
}

void Pentagon::Print() {
```

```
	std::cout << "Point A: " << a_x << ", " << a_y << std::endl;
	std::cout << "Point B: " << b_x << ", " << b_y << std::endl;
	std::cout << "Point C: " << c_x << ", " << c_y << std::endl;
	std::cout << "Point D: " << d_x << ", " << d_y << std::endl;
	std::cout << "Point E: " << e_x << ", " << e_y << std::endl;

}

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

ЛИСТИНГ ФАЙЛА OCTAGON.H

```
#ifndef OCTAGON_H_INCLUDED
#define OCTAGON_H_INCLUDED

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



class Octagon : public Figure {
public:
	Octagon();
	Octagon(std::istream &is);
	Octagon(double i_x, double j_x, double k_x, double l_x, double
m_x, double n_x, double o_x, double p_x, double i_y, double j_y,
double k_y, double l_y, double m_y, double n_y, double o_y, double
p_y);
	Octagon(const Octagon& other);

	double Square() override;
	void   Print() override;
```

```
	virtual ~Octagon();


private:
	double a_x, a_y;
	double b_x, b_y;
	double c_x, c_y;
	double d_x, d_y;
	double e_x, e_y;
	double f_x, f_y;
	double g_x, g_y;
	double h_x, h_y;
};


#endif // OCTAGON_H
```

ЛИСТИНГ ФАЙЛА OCTAGON.CPP

```
#include "Octagon.h"
#include "Triangle.h"
#include <math.h>


Octagon::Octagon() : Octagon(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) {
}


Octagon::Octagon(double i_x, double j_x, double k_x, double l_x, double m_x, double n_x, double o_x, double p_x, double i_y, double j_y, double k_y, double l_y, double m_y, double n_y, double o_y, double p_y) :
	a_x(i_x), a_y(i_y), b_x(j_x), b_y(j_y), c_x(k_x), c_y(k_y), d_x(l_x), d_y(l_y), e_x(m_x), e_y(m_y), f_x(n_x), f_y(n_y), g_x(o_x), g_y(o_y), h_x(p_x), h_y(p_y) {
	std::cout << "Octagon created:  point A: " << a_x << ", " << a_y << std::endl;
	std::cout << "Octagon created:  point B: " << b_x << ", " << b_y << std::endl;
```

```
	std::cout << "Octagon created:  point C: " << c_x << ", " << c_y << std::endl;
	std::cout << "Octagon created:  point D: " << d_x << ", " << d_y << std::endl;
	std::cout << "Octagon created:  point E: " << e_x << ", " << e_y << std::endl;
	std::cout << "Octagon created:  point F: " << f_x << ", " << f_y << std::endl;
	std::cout << "Octagon created:  point G: " << g_x << ", " << g_y << std::endl;
	std::cout << "Octagon created:  point H: " << h_x << ", " << h_y << std::endl;
}

Octagon::Octagon(std::istream &is) {
	is >> a_x >> a_y;
	is >> b_x >> b_y;
	is >> c_x >> c_y;
	is >> d_x >> d_y;
	is >> e_x >> e_y;
	is >> f_x >> f_y;
	is >> g_x >> g_y;
	is >> h_x >> h_y;
}

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

double Octagon::Square() {
	double ab, ac, ad, ah, bc, cd, de, dh, dg, ef, eg, fg, gh;

	ab = sqrt(pow(a_x - b_x, 2) + pow(a_y - b_y, 2));
	ac = sqrt(pow(a_x - c_x, 2) + pow(a_y - c_y, 2));
	ad = sqrt(pow(a_x - d_x, 2) + pow(a_y - d_y, 2));
```

```
	ah = sqrt(pow(a_x - h_x, 2) + pow(a_y - h_y, 2));
	bc = sqrt(pow(c_x - b_x, 2) + pow(c_y - b_y, 2));
	cd = sqrt(pow(c_x - d_x, 2) + pow(c_y - d_y, 2));
	de = sqrt(pow(e_x - d_x, 2) + pow(e_y - d_y, 2));
	dh = sqrt(pow(h_x - d_x, 2) + pow(h_y - d_y, 2));
	dg = sqrt(pow(d_x - g_x, 2) + pow(d_y - g_y, 2));
	ef = sqrt(pow(e_x - f_x, 2) + pow(e_y - f_y, 2));
	eg = sqrt(pow(e_x - g_x, 2) + pow(e_y - g_y, 2));
	fg = sqrt(pow(f_x - g_x, 2) + pow(f_y - g_y, 2));
	gh = sqrt(pow(g_x - h_x, 2) + pow(g_y - h_y, 2));

	double res = 0;

	Triangle *T = new Triangle(ab, ac, bc);
	res += T->Square();
	T->Set(ac, cd, ad);
	res += T->Square();
	T->Set(ah, ad, dh);
	res += T->Square();
	T->Set(dh, dg, gh);
	res += T->Square();
	T->Set(de, eg, dg);
	res += T->Square();
	T->Set(ef, fg, eg);
	res += T->Square();
	delete T;
	return res;
}

Octagon::Octagon(const Octagon& orig) {
	std::cout << "Octagon copy created" << std::endl;
	a_x = orig.a_x;
```

```
	a_y = orig.a_y;
	b_x = orig.b_x;
	b_y = orig.b_y;
	c_x = orig.c_x;
	c_y = orig.c_y;
	d_x = orig.d_x;
	d_y = orig.d_y;
	e_x = orig.e_x;
	e_y = orig.e_y;
	f_x = orig.f_x;
	f_y = orig.f_y;
	g_x = orig.g_x;
	g_y = orig.g_y;
	h_x = orig.h_x;
	h_y = orig.h_y;
}

void Octagon::Print() {
	std::cout << "Point A: " << a_x << ", " << a_y << std::endl;
	std::cout << "Point B: " << b_x << ", " << b_y << std::endl;
	std::cout << "Point C: " << c_x << ", " << c_y << std::endl;
	std::cout << "Point D: " << d_x << ", " << d_y << std::endl;
	std::cout << "Point E: " << e_x << ", " << e_y << std::endl;
	std::cout << "Point F: " << f_x << ", " << f_y << std::endl;
	std::cout << "Point G: " << g_x << ", " << g_y << std::endl;
	std::cout << "Point H: " << h_x << ", " << h_y << std::endl;
}
```

ЛИСТИНГ ФАЙЛА OCTAGON.H

```
#ifndef HEXAGON_H_INCLUDED
#define HEXAGON_H_INCLUDED
```

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



class Hexagon : public Figure {
public:
	Hexagon();
	Hexagon(std::istream &is);
	Hexagon(double i_x, double j_x, double k_x, double l_x, double m_x, double n_x, double i_y, double j_y, double k_y, double l_y, double m_y, double n_y);
	Hexagon(const Hexagon& other);


	double Square() override;
	void   Print() override;


	virtual ~Hexagon();


private:
	double a_x, a_y;
	double b_x, b_y;
	double c_x, c_y;
	double d_x, d_y;
	double e_x, e_y;
	double f_x, f_y;
};


#endif // HEXAGON_H
```

ЛИСТИНГ ФАЙЛА HEXAGON.CPP

```
#include "Hexagon.h"
#include "Triangle.h"
#include <math.h>
```

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


Hexagon::Hexagon(double i_x, double j_x, double k_x, double l_x,
double m_x, double n_x, double i_y, double j_y, double k_y, double
l_y, double m_y, double n_y) :
     a_x(i_x), a_y(i_y), b_x(j_x), b_y(j_y), c_x(k_x), c_y(k_y),
d_x(l_x), d_y(l_y), e_x(m_x), e_y(m_y), f_x(n_x), f_y(n_y) {
     std::cout << "Hexagon created:  point A: " << a_x << ", " <<
a_y << std::endl;
     std::cout << "Hexagon created:  point B: " << b_x << ", " <<
b_y << std::endl;
     std::cout << "Hexagon created:  point C: " << c_x << ", " <<
c_y << std::endl;
     std::cout << "Hexagon created:  point D: " << d_x << ", " <<
d_y << std::endl;
     std::cout << "Hexagon created:  point E: " << e_x << ", " <<
e_y << std::endl;
     std::cout << "Hexagon created:  point F: " << f_x << ", " <<
f_y << std::endl;
}


Hexagon::Hexagon(std::istream &is) {
     is >> a_x >> a_y;
     is >> b_x >> b_y;
     is >> c_x >> c_y;
     is >> d_x >> d_y;
     is >> e_x >> e_y;
     is >> f_x >> f_y;
}


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

```
double Hexagon::Square() {
	double ab, ac, af, bc, cd, cf, de, df, ef;

	ab = sqrt(pow(a_x - b_x, 2) + pow(a_y - b_y, 2));
	ac = sqrt(pow(a_x - c_x, 2) + pow(a_y - c_y, 2));
	af = sqrt(pow(a_x - e_x, 2) + pow(a_y - e_y, 2));
	bc = sqrt(pow(c_x - b_x, 2) + pow(c_y - b_y, 2));
	cd = sqrt(pow(c_x - d_x, 2) + pow(c_y - d_y, 2));
	cf = sqrt(pow(c_x - f_x, 2) + pow(c_y - f_y, 2));
	de = sqrt(pow(e_x - d_x, 2) + pow(e_y - d_y, 2));
	df = sqrt(pow(d_x - f_x, 2) + pow(d_y - f_y, 2));
	ef = sqrt(pow(e_x - f_x, 2) + pow(e_y - f_y, 2));

	double res = 0;

	Triangle *T = new Triangle(ab, ac, bc);
	res += T->Square();
	T->Set(ac, cf, af);
	res += T->Square();
	T->Set(cd, df, cf);
	res += T->Square();
	T->Set(ef, de, df);
	res += T->Square();
	delete T;
	return res;
}

Hexagon::Hexagon(const Hexagon& orig) {
	std::cout << "Hexagon copy created" << std::endl;
	a_x = orig.a_x;
	a_y = orig.a_y;
	b_x = orig.b_x;
```

```
	b_y = orig.b_y;
	c_x = orig.c_x;
	c_y = orig.c_y;
	d_x = orig.d_x;
	d_y = orig.d_y;
	e_x = orig.e_x;
	e_y = orig.e_y;
	f_x = orig.f_x;
	f_y = orig.f_y;
}

void Hexagon::Print() {
	std::cout << "Point A: " << a_x << ", " << a_y << std::endl;
	std::cout << "Point B: " << b_x << ", " << b_y << std::endl;
	std::cout << "Point C: " << c_x << ", " << c_y << std::endl;
	std::cout << "Point D: " << d_x << ", " << d_y << std::endl;
	std::cout << "Point E: " << e_x << ", " << e_y << std::endl;
	std::cout << "Point F: " << f_x << ", " << f_y << std::endl;
}
```

Выводы:

В ходе выполнения лабораторной работы был получен навык работы с классами в C++. Я научился создавать классы и использовать объекты этих классов. Был спроектирован и запрограммирован на языке C++ класс фигуры: квадрат. Так же были изучены основные понятия ООП и ознакомление с перегрузкой операторов.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №2

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №2

Цели:

- Закрепление навыков работы с классами.
- Создание простых динамических структур данных.
- Работа с объектами, передаваемыми «по значению».

Задание:

Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий одну фигуру (колонка фигура 1), согласно вариантам заданий (реализованную в ЛР1).

Контейнер первого уровня в моём варианте – массив.

Классы должны удовлетворять следующим правилам:

- Требования к классу фигуры аналогичны требованиям из лабораторной работы 1.
- Классы фигур должны иметь переопределенный оператор вывода в поток std::ostream (<<). Оператор должен распечатывать параметры фигуры (тип фигуры, длины сторон, радиус и т.д).
- Классы фигур должны иметь переопределенный оператор ввода фигуры из потока std::istream (>>). Оператор должен вводить основные параметры фигуры (длины сторон, радиус и т.д).
- Классы фигур должны иметь операторы копирования (=).
- Классы фигур должны иметь операторы сравнения с такими же фигурами (==).
- Класс-контейнер должен соджержать объекты фигур “по значению” (не по ссылке).
- 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).

Код:

ЛИСТИНГ ФАЙЛА MASSIVE.H

```
#ifndef MASSIVE_H
#define MASSIVE_H

#include "pentagon.h"

class Massive
{
private:
	Pentagon *massive_Body;
	unsigned int massive_Size;
	unsigned int massive_Capacity;

public:
	Massive();

	bool Push_back(Pentagon &data);
	bool empty();
	Pentagon Pop_back();
	int Size() const;
	bool Set(int pos, Pentagon &data);
	Pentagon Get(int pos)const;

	friend std::ostream& operator<<(std::ostream& os, const
Massive& massive);
	Pentagon operator [](int index);

	virtual ~Massive();
};

#endif // MASSIVE_H
```

ЛИСТИНГ ФАЙЛА MASSIVE.CPP

```
#include "Massive.h"
#include "pentagon.h"


Massive::Massive() {
	this->massive_Capacity = 10;
	this->massive_Size = 0;
	this->massive_Body = (Pentagon*)malloc(this->massive_Capacity *
sizeof(Pentagon));
}


Pentagon Massive::operator [](int index) {
	return this->massive_Body[index];
}


bool Massive::Set(int pos, Pentagon &data) {
	if (pos >= this->massive_Size) {
		return false;
	}
	this->massive_Body[pos] = data;;
	return true;
}


Pentagon Massive::Get(int pos) const {
	return (this->massive_Body[pos]);
}


bool Massive::Push_back(Pentagon &data) {
	if (this->massive_Size + 1 > this->massive_Capacity) {
		this->massive_Capacity = this->massive_Capacity * 1.5;
		this->massive_Body = (Pentagon*)realloc(this-
>massive_Body, this->massive_Capacity * sizeof(Pentagon));
	}
	if (this->massive_Body == nullptr) {
```

```
            free(this->massive_Body);
            this->massive_Capacity = 0;
            this->massive_Size = 0;
            return false;
      }
      this->massive_Body[this->massive_Size] = Pentagon(data);
      ++this->massive_Size;
      return true;
}


Pentagon Massive::Pop_back() {
      Pentagon backtrack = this->massive_Body[this->massive_Size -
1];
      this->massive_Size -= 1;
      this->massive_Body = (Pentagon*)realloc(this->massive_Body,
this->massive_Size * sizeof(Pentagon));
      this->massive_Capacity = this->massive_Size;
      return backtrack;
}
bool Massive::empty() {
      return this->massive_Size == 0;
}


int Massive::Size() const {
      return this->massive_Size;
}
std::ostream& operator<<(std::ostream& os, const Massive& massive) {
      for (int i = 0; i < massive.Size(); i++) {
            Pentagon p = massive.Get(i);
            os << p;
      }
}
Massive::~Massive() {
```

```
	free(this->massive_Body);
}
```

Выводы:

В ходе этой лабораторной работы были закреплены навыки работы с классами. Повторены способы создания и использования объектов этих классов. Так же я ознакомился с динамическими структурами. Были написаны динамические структуры работающие с объектами, передаваемыми “по значению”.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №3

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №3

Цели:

- 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 вариантам заданий (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).

Код:

ЛИСТИНГ ФАЙЛА MAIN.CPP

```
# #include <cstdlib>
#include <memory>
#include "triangle.h"
#include "pentagon.h"
//#include "Massive.h"
#include "Container.h"
#include "Hexagon.h"
```

```
#include "Octagon.h"



int main() {



	char command;


	//std::cout << "Please, select the figure you want to calculate:" << std::endl;

	//std::cout << "use p to select Pentagon, h to select Hexagon, o to select Octagon, q to Quit" << std::endl;

	std::cout << "Please, fill in a vector of figures." << std::endl;

	std::cout << "Use p to push a new figure at the back of a vector.\nUse a to add a figure at your desired position." << std::endl;

	std::cout << "Use s to get the size of the vector.\nUse o to print the vector from output stream.\nUse q to quit." << std::endl;

	std::cout << "Use d to delete a figure from the back.\n\nPlease, note, that you must only use indexes that are smaller than the size of the vector!" << std::endl;

	std::cin >> command;


	//Massive* my_container = new Massive();

	//Figure *Fig_ptr;

	Container* my_container = new Container();

	std::shared_ptr<Figure> my_figure = nullptr;


	while (command != 'q') {


		if (command == 'a') {

			int pos;

			std::cout << "Please, type in the position of your figure!" << std::endl;

			std::cin >> pos;
```

```
                if (pos >= my_container->Size()) {
                    std::cout << "You have tried accessing a
nonexistent field! Quitting the program." << std::endl;
                    return 1;
                }

                std::cout << "Please type the first letter of the
type of your figure." << std::endl;
                std::cin >> command;
                if (command == 'p') {
                    std::cout << "You chose to add a pentagon!" <<
std::endl;

                    std::cout << "Please, type in five points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Pentagon(std::cin));
                }
                if (command == 'h') {
                    std::cout << "You chose to add a hexagon!" <<
std::endl;

                    std::cout << "Please, type in six points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Hexagon(std::cin));
                }
                if (command == 'o') {
                    std::cout << "You chose to add a octagon!" <<
std::endl;

                    std::cout << "Please, type in eight points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Octagon(std::cin));
```

```
                }


                my_container->Set(pos, my_figure);
                //break;
           }


           else if (command == 'p') {
                std::cout << "Please type the first letter of the
type of your figure." << std::endl;
                std::cin >> command;
                if (command == 'p') {
                     std::cout << "You chose to add a pentagon!" <<
std::endl;


                     std::cout << "Please, type in five points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                     my_figure = std::shared_ptr<Figure>(new
Pentagon(std::cin));
                }
                if (command == 'h') {
                     std::cout << "You chose to add a hexagon!" <<
std::endl;


                     std::cout << "Please, type in six points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                     my_figure = std::shared_ptr<Figure>(new
Hexagon(std::cin));
                }
                if (command == 'o') {
                     std::cout << "You chose to add a octagon!" <<
std::endl;


                     std::cout << "Please, type in eight points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
```

```
                    my_figure = std::shared_ptr<Figure>(new
Octagon(std::cin));
                }


                if (!(my_container->Push_back(my_figure))) {
                    std::cout << "Something went wrong! Quitting
the program." << std::endl;
                    return 1;
                }


            }


            else if (command == 's') {
                std::cout << "The size of your vector is " <<
my_container->Size() << std::endl;
            }


            else if (command == 'o') {
                std::cout << *my_container;
            }


            else if (command == 'd') {
                if (!(my_container->empty())) {
                    my_container->Pop_back();
                    std::cout << "Deleted the last element from
the vector." << std::endl;
                }
                else {
                    std::cout << "Trying to delete from an empty
vector!" << std::endl;
                }
            }
            else {
                std::cout << "Wrong command!" << std::endl;
```

```
            }
            my_figure = nullptr;
            std::cout << "Please, enter your next command: ";
            std::cin >> command;
      }
      delete my_container;
      return 0;
}
```

ЛИСТИНГ ФАЙЛА CONTAINER.H

```
#ifndef CONTAINER_H
#define CONTAINER_H

#include "Figure.h"
#include <memory>

class Container
{

private:
      std::shared_ptr<Figure>*  massive_Body;
      unsigned int massive_Size;
      unsigned int massive_Capacity;

public:
      Container();

      bool Push_back(std::shared_ptr<Figure> &data);
      bool empty();
      std::shared_ptr<Figure> Pop_back();
      int Size() const;
      bool Set(int pos, std::shared_ptr<Figure> &data);
      std::shared_ptr<Figure> Get(int pos)const;
```

```
	friend std::ostream& operator<<(std::ostream& os, const
Container& massive);
	std::shared_ptr<Figure> operator [](int index);



	~Container();

protected:

private:
};

#endif // CONTAINER_H
```

ЛИСТИНГ ФАЙЛА CONTAINER.CPP

```
#include "Container.h"
#include "Pentagon.h"

Container::Container() {
	this->massive_Capacity = 10;
	this->massive_Size = 0;
	this->massive_Body = new std::shared_ptr<Figure>[10];
}

std::shared_ptr<Figure> Container::operator [](int index) {
	return this->massive_Body[index];
}

bool Container::Set(int pos, std::shared_ptr<Figure> &data) {
	if (pos >= this->massive_Size) {
		return false;
	}
```

```
	this->massive_Body[pos - 1] = data;;
	return true;
}

std::shared_ptr<Figure> Container::Get(int pos) const {
	return (this->massive_Body[pos]);
}

bool Container::Push_back(std::shared_ptr<Figure> &data) {
	if (this->massive_Size == this->massive_Capacity) {
		this->massive_Capacity = this->massive_Capacity * 1.5;
		std::shared_ptr<Figure> new_body[this->massive_Capacity];
		for (int i = 0; i < this->massive_Size; i++) {
			new_body[i] = this->massive_Body[i];
			//free(this->massive_Body[i]);
		}
		this->massive_Body = new_body;
	}
	if (this->massive_Body == nullptr) {
		free(this->massive_Body);
		this->massive_Capacity = 0;
		this->massive_Size = 0;
		return false;
	}
	this->massive_Body[this->massive_Size] = data;
	++this->massive_Size;
	return true;
}
std::shared_ptr<Figure> Container::Pop_back() {
	std::shared_ptr<Figure> backtrack = this->massive_Body[this-
>massive_Size - 1];
	this->massive_Body[this->massive_Size - 1] = nullptr;
	this->massive_Size -= 1;
```

```
	//this->massive_Body = (std::shared_ptr<Figure>)realloc(this->massive_Body, this->massive_Size * sizeof(std::shared_ptr<Figure>));
	//this->massive_Capacity = this->massive_Size;
	return backtrack;
}
bool Container::empty() {
	return this->massive_Size == 0;
}
int Container::Size() const {
	return this->massive_Size;
}
std::ostream& operator<<(std::ostream& os, const Container& massive)
{
	for (int i = 0; i < massive.Size(); i++) {
		os << i << "th element is:\n";
		std::shared_ptr<Figure> p = massive.Get(i);
		p->Print();
		os << '\n';
		os << '\n';
	}
	return os;
}
Container::~Container() {
}
```

Выводы:

В ходе выполнения данной лабораторной работы были окончательно закреплены навыки работы с классами и был получен навык использования “умных” указателей. Были спроектированы и запрограммированы на языке C++ классы фигур: квадрат, прямоугольник и трапеция. Все программы проекта были переписаны на использование “умных” указателей.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №4

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №4

Цели:

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

Задание:

Необходимо спроектировать и запрограммировать на языке C++ шаблон класса-контейнера первого уровня, содержащий все три фигуры класса фигуры, согласно вариантам заданий (реализованную в ЛР1).

Классы должны удовлетворять следующим правилам:

- Требования к классам фигуры аналогичны требованиям из лабораторной работы 1.
- Шаблон класса-контейнера должен соджержать объекты используя std:shared_ptr<…>.
- Шаблон класса-контейнера должен иметь метод по добавлению фигуры в контейнер.
- Шаблон класса-контейнера должен иметь методы по получению фигуры из контейнера (опередеяется структурой контейнера).
- Шаблон класса-контейнера должен иметь метод по удалению фигуры из контейнера (опередеяется структурой контейнера).
- Шаблон класса-контейнера должен иметь перегруженный оператор по выводу контейнера в поток std::ostream (<<).
- Шаблон класса-контейнера должен иметь деструктор, удаляющий все элементы контейнера.
- Классы должны быть расположенны в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Код:

ЛИСТИНГ ФАЙЛА CONTAINER.H

```
// *** ADDED BY HEADER FIXUP ***
#include <istream>
// *** END ***
#ifndef CONTAINER_H
#define CONTAINER_H


#include <memory>
#include <iostream>
```

```
#include "TIterator.h"
//#include "TAllocator.h"

template <class T>
class Container
{

    private:
        std::shared_ptr<T>*  massive_Body;
        unsigned int massive_Size;
        unsigned int massive_Capacity ;
        //static inline TAllocator container_alloc;

    public:
        Container(){
            this->massive_Capacity = 10;
            this->massive_Size = 0;
            this->massive_Body = new std::shared_ptr<T>[10];
        }

        //TAllocator container_alloc(sizeof(T), 100);

        bool Push_back (std::shared_ptr<T> &data){
            if (this->massive_Size == this->massive_Capacity){
                this->massive_Capacity = this->massive_Capacity *
1.5;
                std::shared_ptr<T> new_body [this-
>massive_Capacity];
                for (int i = 0; i < this->massive_Size; i++){
                    new_body[i] = this->massive_Body[i];
                    //free(this->massive_Body[i]);
                }
                this->massive_Body = new_body;
```

```
            }
            if (this->massive_Body == nullptr){
                free(this->massive_Body);
                this->massive_Capacity = 0;
                this->massive_Size = 0;
                return false;
            }
            this->massive_Body[this->massive_Size] = data;
            ++this->massive_Size;
            return true;
        }

        bool empty(){
            return this->Size() == 0;
        }

        std::shared_ptr<T> Pop_back(){
            std::shared_ptr<T> backtrack = this->massive_Body[this->massive_Size-1];
            this->massive_Body[this->massive_Size-1] = nullptr;
            this->massive_Size -= 1;
            //this->massive_Body = (std::shared_ptr<T>)realloc(this->massive_Body, this->massive_Size * sizeof(std::shared_ptr<T>));
            //this->massive_Capacity = this->massive_Size;
            return backtrack;
        }


        int Size() const {
            return this->massive_Size;
        }

        bool Set(int pos, std::shared_ptr<T> &data){
```

```
            if(pos >= this->massive_Size){
                return false;
            }
            this->massive_Body[pos-1] = data;;
            return true;
        }

        std::shared_ptr<T> Get(int pos)const {
            return (this->massive_Body[pos]);
        }

        friend std::ostream& operator << (std::ostream& os,const
Container<T>* massive){
            for (int i = 0; i < massive->Size(); i++){
                std::cout << i << "th element is:\n";
                std::shared_ptr<T> p = massive->Get(i);
                p->Print();
                std::cout << "\n\n";
            }
            return os;
        }

        TIterator <T> begin(){
            return TIterator<T>(massive_Body,0);
        }
        TIterator <T> end(){
            if (this->Size() > 0){
                return TIterator<T>(massive_Body,this->Size());
            }
            return TIterator<T>(nullptr, 0);

        }
```

```
        ~Container(){
            for (int i = 0; i < this->Size(); i++){
                this->massive_Body[i] = nullptr;
            }
        }
};


#endif // CONTAINER_H
```

Выводы:

В ходе выполнения лабораторной работы был получен навык работы с шаблонами в C++. Я научился создавать шаблоны классов и использовать их. Была спроектирована и запрограммирована на языке C++ динамическая структура, использующая шаблоны ранее запрограммированных классов.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №5

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №5

Цели:

- Закрепление навыков работы с шаблонами классов.
- Построение итераторов для динамических структур данных.

Задание:

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;
```

Код:

ЛИСТИНГ ФАЙЛА MAIN.CPP

```
// #include <cstdlib>

#include "Triangle.h"

#include "Pentagon.h"

#include "TContainer.h"

#include "Hexagon.h"

#include "Octagon.h"

#include "TIterator.h"



int main() {



    char command;
```

```
    std::cout << "Please, fill in a vector of figures." <<
std::endl;

    std::cout << "Use p to push a new figure at the back of a
vector.\nUse a to add a figure at your desired position." <<
std::endl;

    std::cout << "Use s to get the size of the vector.\nUse o to
print the vector from output stream.\nUse q to quit." << std::endl;

    std::cout << "Use i to call iterative output." << std::endl;

    std::cout << "Use d to delete a figure from the back.\n\nPlease,
note, that you must only use indexes that are smaller than the size
of the vector!" << std::endl;

    std::cin >> command;


    TContainer<Figure>* my_TContainer;

    std::shared_ptr<Figure> my_figure = nullptr;


    my_TContainer = new TContainer<Figure>();


    while (command != 'q'){


            if (command == 'a') {

                int pos;

                std::cout << "Please, type in the position of your
figure!" << std::endl;

                std::cin >> pos;

                if( pos >= my_TContainer->Size()) {

                        std::cout << "You have tried accessing a
nonexistent field! Quitting the program." << std::endl;

                        return 1;

                }


                std::cout << "Please type the first letter of the
type of your figure." << std::endl;

                std::cin >> command;

                if (command == 'p'){
```

```
                    std::cout << "You chose to add a pentagon!" <<
std::endl;


                    std::cout << "Please, type in five points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Pentagon(std::cin));
                }
                if (command == 'h'){
                    std::cout << "You chose to add a hexagon!" <<
std::endl;


                    std::cout << "Please, type in six points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Hexagon(std::cin));
                }
                if (command == 'o'){
                    std::cout << "You chose to add a octagon!" <<
std::endl;


                    std::cout << "Please, type in eight points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Octagon(std::cin));
                }


                my_TContainer->Set(pos, my_figure);
                //break;
            }


            else if (command == 'p'){
                std::cout << "Please type the first letter of the
type of your figure." << std::endl;
```

```
                std::cin >> command;
                if (command == 'p'){
                    std::cout << "You chose to add a pentagon!" <<
std::endl;

                    std::cout << "Please, type in five points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Pentagon(std::cin));
                }
                if (command == 'h'){
                    std::cout << "You chose to add a hexagon!" <<
std::endl;

                    std::cout << "Please, type in six points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Hexagon(std::cin));
                }
                if (command == 'o'){
                    std::cout << "You chose to add a octagon!" <<
std::endl;
                    std::cout << "Please, type in eight points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Octagon(std::cin));
                }

                if(!(my_TContainer->Push_back(my_figure))){
                    std::cout << "Something went wrong! Quitting the
program." << std::endl;
                    return 1;
                }
            }
```

```
            else if (command == 's'){
                std::cout << "The size of your vector is " <<
my_TContainer->Size() << std::endl;
            }
            else if (command == 'o'){
                std::cout << my_TContainer;
            }
            else if (command == 'd'){
                if (!(my_TContainer->empty())){
                    my_TContainer->Pop_back();
                    std::cout << "Deleted the last element from the
vector." << std::endl;
                }
                else {
                    std::cout << "Do not try this at home!\nTrying
to delete from an empty vector gone wrong!" << std::endl;
                }
            }
            else if (command == 'i'){
                std::cout << "Printing with iterators." <<
std::endl;
                for (auto i : *my_TContainer){
                    std::cout << "Printing element:" << std::endl;
                    i->Print();
                }
            }
            else {
                    std::cout << "Wrong command!" << std::endl;
            }
            my_figure = nullptr;
            std::cout << "Please, enter your next command: ";
            std::cin >> command;
    }
    delete my_TContainer;
```

```
    return 0;
}
```

ЛИСТИНГ ФАЙЛА TITERATOR.H

```
#ifndef TITERATOR_H_INCLUDED
#define TITERATOR_H_INCLUDED

//#include "TContainer.h"

template <class T>
class TIterator
{
public:

    TIterator(std::shared_ptr<T>* n, int p)   {

        my_Body = n;
        pos = p;
    }
    std::shared_ptr<T> operator * (){
        return my_Body[pos];
    }
    std::shared_ptr<T> operator -> (){
        return my_Body[pos];
    }
    void operator ++ (){
        pos += 1;
    }
    TIterator operator ++ (int){
        TIterator iter(*this);
        ++(*this);
        return iter;
```

```
    }
    bool operator == (TIterator const& i){
        return (my_Body == i.my_Body) && (pos == i.pos);
    }

    bool operator != (TIterator const& i){
        return !(*this == i);
    }
private:
    std::shared_ptr<T>*  my_Body;
    int pos;
};
#endif // TITERATOR_H_INCLUDED
```

Выводы:

В ходе выполнения лабораторной работы был получен навык работы с итераторами в C++. Я научился создавать итераторы классов и использовать итераторы объектов этих классов. Были спроектированы и запрограммированы на языке C++ итераторы классов фигур: квадрат, прямоугольник и трапеция. Также были закреплены основы работы с шаблоными.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №6

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №6

Цели:

- Закрепление навыков по работе с памятью в C++.
- Создание аллокаторов памяти для динамических структур данных.

Задание:

Используя структуры данных, разработанные для предыдущей лабораторной работы (ЛР№5) спроектировать и разработать аллокатор памяти для динамической структуры данных.

Цель построения аллокатора – минимизация вызова операции malloc. Аллокатор должен выделять большие блоки памяти для хранения фигур и при создании новых фигур-объектов выделять место под объекты в этой памяти.

Алокатор должен хранить списки использованных/свободных блоков. Для хранения списка свободных блоков нужно применять динамическую структуру данных (контейнер 2-го уровня, согласно вариантам заданий). В моём случае – стек.

Для вызова аллокатора должны быть переопределены оператор new и delete у классов-фигур.

Код:

ЛИСТИНГ ФАЙЛА TALLOCATOR.H

```
#ifndef TALLOCATOR_H_INCLUDED
#define TALLOCATOR_H_INCLUDED

#include <cstdlib>
#include "TStack.h"



class TAllocator {
public:
```

```
    TAllocator(size_t size,size_t count);
    void *allocate();
    void deallocate(void *pointer);
    bool has_free_blocks();
    virtual ~TAllocator();
private:
    size_t _size;
    size_t _count;

    char  *_used_blocks;
    TStack<void>* _free_blocks;

    size_t _free_count;
};
#endif // TALLOCATOR_H_INCLUDED
```

ЛИСТИНГ ФАЙЛА TALLOCATOR.CPP

```
#include "TAllocator.h"
#include "TStack.h"
#include <iostream>



TAllocator::TAllocator(size_t size,size_t count):
_size(size),_count(count)  {
    _used_blocks = (char*)malloc(_size*_count);

    //_free_blocks = (void**)malloc(sizeof(void*)*_count);
    _free_blocks = new TStack<void>();
    for (size_t i = 0; i < count; i++){
```

```
        _free_blocks->Push(_used_blocks + i*_size);
    }

    //for(size_t i=0;i<_count;i++) _free_blocks[i] =
_used_blocks+i*_size;
    _free_count = _count;
      std::cout << "TAllocator: Memory init" << std::endl;
}

void *TAllocator::allocate() {
    void *result = nullptr;

    if(_free_count>0)
    {
        result = _free_blocks->Pop();
        _free_count--;
        std::cout << "TAllocator: Allocate " << (_count-_free_count)
<< " of " << _count << std::endl;
    } else
    {
        std::cout << "TAllocator: No memory exception" << std::endl;
    }

    return result;
}

void TAllocator::deallocate(void *pointer) {
  std::cout << "TAllocator: Deallocate block "<< std::endl;

  _free_blocks->Push(pointer);
  _free_count ++;

}
```

```
bool TAllocator::has_free_blocks() {
    return _free_count>0;
}
TAllocator::~TAllocator() {
    if(_free_count<_count) std::cout << "TAllocator: Memory leak?"
<< std::endl;
                    else  std::cout << "TAllocator: Memory freed" <<
std::endl;
    delete _free_blocks;
    delete _used_blocks;
}
```

ЛИСТИНГ ФАЙЛА TSTACK.H

```
#ifndef TSTACK_H_INCLUDED
#define TSTACK_H_INCLUDED

#include "TStackElem.h"

template <class T>

class TStack{

private:
   TStackElem* head;

public:
    TStack (){
        head = nullptr;
    }

    void Push (T* data){
        TStackElem* newel = new TStackElem(head,(void*) data);
```

```
        head = newel;
    }
    T* Pop (){
        TStackElem* oldel = head;
        head = oldel->next;
        T* data = (T*)oldel->data;
        delete oldel;
        return data;
    }
    ~ TStack(){
        while (head != nullptr){
            this->Pop();
        }
    }
};
#endif // TSTACK_H_INCLUDED
```

ЛИСТИНГ ФАЙЛА TSTACKELEM.H

```
#ifndef TSTACKELEM_H_INCLUDED
#define TSTACKELEM_H_INCLUDED

class TStackElem {
public:
    void* data;
    TStackElem* next;
    TStackElem(){
        data = nullptr;
        next = nullptr;
    }
    TStackElem(TStackElem* prev, void* input){
        next = prev;
        data = input;
```

```
    }
    ~TStackElem(){
        //DELETING
    }
};
#endif // TSTACKELEM_H_INCLUDED
```

Выводы:

В ходе выполнения лабораторной работы был получен навык работы с аллокаторами классов в C++. Я научился писать алкллокаторы классов. Был спроектирован и запрограммирован на языке C++ аллокатор динамического класса, использующий другой динамический класс в основе. Так же были закреплены основные понятия о работе с памятью в C++.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №7

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №7

Цели:

- Создание сложных динамических структур данных.
- Закрепление принципа ООП.

Задание:

Необходимо реализовать динамическую структуру данных – «Хранилище объектов» и алгоритм работы с ней. «Хранилище объектов» представляет собой контейнер, одного из следующих видов (Контейнер 1-го уровня):

1. Массив
2. Связный список
3. Бинарное- Дерево.
4. N-Дерево (с ограничением не больше 4 элементов на одном уровне).
5. Очередь
6. Стек

Каждым элементом контейнера, в свою, является динамической структурой данных одного из следующих видов (Контейнер 2-го уровня):

1. Массив
2. Связный список
3. Бинарное- Дерево
4. N-Дерево (с ограничением не больше 4 элементов на одном уровне).
5. Очередь
6. Стек

Таким образом у нас получается контейнер в контейнере.

Элементом второго контейнера является объект-фигура, определенная вариантом задания.

При этом должно выполняться правило, что количество объектов в контейнере второго уровня не больше 5. Объекты в контейнерах второго уровня должны быть отсортированы по возрастанию площади объекта (в том числе и для деревьев).

При удалении объектов должно выполняться правило, что контейнер второго уровня не должен быть пустым. Т.е. если он становится пустым, то он должен удалится.

Код:

ЛИСТИНГ ФАЙЛА TCONTAINER.H

```
#ifndef TCONTAINER_H_INCLUDED
#define TCONTAINER_H_INCLUDED
#include <memory>
#include <iostream>
#include "TIterator.h"
#include "MyStack.h"
#include "TFigure.h"

class TContainer
{
    private:
        MyStack <std::shared_ptr<Figure>>*  massive_Body;
        unsigned int massive_Size;
        unsigned int massive_Capacity ;
    public:
        TContainer(){
            this->massive_Capacity = 10;
            this->massive_Size = 0;
            this->massive_Body = new MyStack
<std::shared_ptr<Figure>>[10];
        }
        bool Push_back (std::shared_ptr<Figure> data){
            if (this->massive_Size == this->massive_Capacity){
                massive_Capacity *= 1.5;
                MyStack<std::shared_ptr<Figure>>* newbody = new
MyStack<std::shared_ptr<Figure>>[massive_Capacity];
                if (!newbody){
                    return false;
                }
                for(int i = 0; i < massive_Size; i++){
                    newbody[i] = massive_Body[i];
                }
                delete massive_Body;
```

```
                massive_Body = newbody;
            }
MyStack<std::shared_ptr <Figure> >();
            massive_Body[massive_Size].Push(data);
            ++massive_Size;
            return true;
        }

        bool empty(){
            return this->Size() == 0;
        }
        void del(char t){
            for (int i = 0; i < massive_Size; i++){
                std::cout << "Checking " << i << "th stack.\n";
                massive_Body[i].del(t);
                std::cout << "Checking " << i << "th stack is
finished.\n";
                if (massive_Body[i].size() == 0){
                    for (int j = i+1; j < massive_Size; j++){
                        massive_Body[j-1] = massive_Body[j];
                    }
                    --this->massive_Size;
                }
            }
        }

        void del(double sq){
            for (int i = 0; i < massive_Size; i++){
                std::cout << "Checking " << i << "th stack.\n";
                massive_Body[i].del(sq);
                if (massive_Body[i].size() == 0){
                    for (int j = i+1; j < massive_Size; j++){
                        massive_Body[j-1] = massive_Body[j];
```

```
                    }
                    --this->massive_Size;
                }
            }
        }
        std::shared_ptr<Figure> Pop_back(){
            return massive_Body[massive_Size - 1].Pop();
        }
        int Size() const {
            return this->massive_Size;
        }
        bool Set(int pos, std::shared_ptr<Figure> &data){
            if(this->massive_Body[pos].size() < 5) {
                    std::cout << "Pushing at " << pos << ".\n";
                    massive_Body[pos].Push(data);
            }
            else {
                std::cout << "The stack you requested is already
full!\n Type c to continue with adding anyway, type anything else to
stop." << std::endl;
                char c;
                std::cin >> c;
                if (c != 'c'){
                    return true;
                }
                else{
                    int cp = pos;
                    while (cp < massive_Size){
                        if(this->massive_Body[cp].size() < 5) {
                            massive_Body[cp].Push(data);
                            return true;
                        }
                        ++cp;
```

```
                    }
                    std::cout << "The massive is full, pushing a new
element." << std::endl;
                    this->Push_back(data);
                }
            }
            return true;
        }
        MyStack <std::shared_ptr<Figure>>* Get(int pos)const {
            return &(this->massive_Body[pos]);
        }
        friend std::ostream& operator << (std::ostream& os,const
TContainer* massive){
            std::cout << "Printing massive." << std::endl;
            for (int i = 0; i < massive->Size(); i++){
                std::cout << i << "th stack is:\n";
                MyStack <std::shared_ptr<Figure>>* p = massive-
>Get(i);
                p->Print();
                std::cout << "\n\n";
            }
            return os;
        }
        TIterator <std::shared_ptr<Figure>> begin(){
            return
TIterator<std::shared_ptr<Figure>>(massive_Body,0);
        }
        TIterator <std::shared_ptr<Figure>> end(){
            if (this->Size() > 0){
                return
TIterator<std::shared_ptr<Figure>>(massive_Body,this->Size());
            }
            return TIterator <std::shared_ptr<Figure>>(nullptr, 0);
        }
```

```
        ~TContainer(){
            for (int i = 0; i < this->Size(); i++){
                while (massive_Body[i].size() > 0){
                    massive_Body[i].Pop();
                }
            }
            delete this->massive_Body;
        }
};
#endif // TContainer_H_INCLUDED
```

ЛИСТИНГ ФАЙЛА TITERATOR.H

```
#ifndef TITERATOR_H_INCLUDED
#define TITERATOR_H_INCLUDED
#include "MyStack.h"
template <class T>
class TIterator
{
public:
    TIterator(MyStack<T>* n, int p)   {
        my_Body = n;
        pos = p;
    }
    MyStack<T>* operator * (){
        return &(my_Body[pos]);
    }
    MyStack<T>* operator -> (){
        return my_Body[pos];
    }
    void operator ++ (){
        ++pos;
    }
```

```
    TIterator operator ++ (int){
        TIterator iter(*this);
        ++(*this);
        return iter;
    }
    bool operator == (TIterator const& i){
        return (my_Body == i.my_Body) && (pos == i.pos);
    }
    bool operator != (TIterator const& i){
        return !(*this == i);
    }
private:
    MyStack<T>*  my_Body;
    int pos;
    //int st_pos;
};
#endif // TITERATOR_H_INCLUDED
```

ЛИСТИНГ ФАЙЛА MYSTACKELEM.H

```
#ifndef MYSTACKELEM_H_INCLUDED
#define MYSTACKELEM_H_INCLUDED

#include <cstdlib>
#include <memory>

template <class T>
class MyStackElem {
private:
    T data;
    std::shared_ptr<MyStackElem<T>> next;
public:
    MyStackElem(){
```

```
        data = nullptr;
        next = nullptr;
    }
    MyStackElem(std::shared_ptr<MyStackElem<T>> prev, T input){
        next = prev;
        data = input;
    }
    std::shared_ptr<MyStackElem<T>> GetNext(){
        return next;
    }
    void SetNext (std::shared_ptr<MyStackElem<T>> newnext){
        next = newnext;
        return;
    }
    void Print(){
        data->Print();
        return;
    }
    T Val(){
        return data;
    }
    ~MyStackElem(){
        //DELETING
    }
};
#endif // MYSTACKELEM_H_INCLUDED
```

ЛИСТИНГ ФАЙЛА MYSTACK.H

```
#ifndef MYSTACK_H_INCLUDED
#define MYSTACK_H_INCLUDED
#include "MyStackElem.h"


template <class T>
```

```
class MyStack{
public:
    int msize;
    std::shared_ptr<MyStackElem <T>> head;
    MyStack (){
        head = nullptr;
        msize = 0;
    }
    void Push (T data){
        std::shared_ptr<MyStackElem <T>> newel =
std::shared_ptr<MyStackElem <T>>(new MyStackElem<T>(head, data));
        std::shared_ptr<MyStackElem <T>> leftel = head;
        if (!head || ((data->Square()) >= (head->Val()->Square()))){
            head = newel;
            std::cout << "Head set" << std::endl;
        }
        else {
            std::shared_ptr<MyStackElem <T>> rightel = head-
>GetNext();
            T right;
            while (rightel && (data->Square() >= right->Square())){
                    right = rightel->Val();
                    rightel = rightel->GetNext();
                    leftel = leftel->GetNext();
                    right = rightel->Val();

            }
            if (rightel){
                newel->SetNext(rightel);
            }
            leftel->SetNext(newel);
        }
        ++msize;
```

```
    }
    T Pop (){
        std::shared_ptr<MyStackElem <T>> oldel = head;
        head = oldel->GetNext();
        T data = oldel->Val();
        oldel = nullptr;
        --msize;
        return data;
    }
    T Peak(){
        return  head->Val();
    }
    void del(char t){
        std::shared_ptr<MyStackElem <T>> ml = head;
        std::shared_ptr<MyStackElem <T>> rl;
        if (!ml){
            std::cout << "Empty stack!" << std::endl;
            return;
        }
        std::cout << "Checking head: type '" << ml->Val()->type <<
"'. \n";
        while (ml->Val()->type == t){
            head = ml->GetNext();
            rl = ml->GetNext();
            ml = nullptr;
            std::cout << "Element deleted.\n";
            if (rl == nullptr){
                std::cout << "Stack devastated!\n";
                msize = 0;
                return;
            }
            --msize;
            ml = rl;
```

```
        }
        while (ml->GetNext()){
            rl = ml->GetNext();
            T rl_val = rl->Val();
            if (rl_val->type == t){
                ml->SetNext(rl->GetNext());
                rl = nullptr;
                std::cout << "Element deleted.\n";
                continue;
            }
            ml = rl;
        }
    }
    void del(double sq){
        std::shared_ptr<MyStackElem <T>> ml = head;
        std::shared_ptr<MyStackElem <T>> rl;
        if (!ml){
            std::cout << "Empty stack!" << std::endl;
            return;
        }
        double t = 1;
        for (int i = 0; i < 16; i++){
            t /= 2;
        }
        std::cout << "Checking head: square = " << ((ml->Val()->Square() - sq) < t) << "\n";
        while (ml && (ml->Val()->Square() - sq) < t){
            head = ml->GetNext();
            rl = ml->GetNext();
            ml = nullptr;
            std::cout << "Element deleted.\n";
            if (rl == nullptr){
                std::cout << "Stack devastated!\n";
```

```
                msize = 0;
                return;
            }
            --msize;
            ml = rl;
        }
        while (ml->GetNext()){
            rl = ml->GetNext();
            T rl_val = rl->Val();
            if (rl_val->Square() - sq < t){
                ml->SetNext(rl->GetNext());
                rl = nullptr;
                --msize;
                std::cout << "Element deleted.\n";
                continue;
            }
            ml = rl;
        }
    }
    const int size(){
        return msize;
    }
    void Print() {
        std::shared_ptr<MyStackElem <T>> focus = head;
        if (!focus){
            std::cout << "Empty stack!" << std::endl;
        }
        while (focus != nullptr){
            std::cout << "Stack Element:\n";
            focus->Print();
            focus = focus->GetNext();
        }
```

```
        return;
    }
    MyStack& operator = (const MyStack& orig){
        this->head = nullptr;
        this->head = orig.head;
        this->msize = orig.msize;
        return *this;
    }
    ~ MyStack(){
        while (head != nullptr){
            this->Pop();
        }
    }
};
#endif // MYSTACK_H_INCLUDED
```

Выводы:

В ходе выполнения лабораторной работы был значительно повышен навык программирования динамических структур. Была спроектирована и написана динамическая структура, имеющая внутри себя объекты другой динамической структуры. Были закреплены основные принципы ООП.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №8

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №8

Цели:

- Знакомство с параллельным программированием в C++.

Задание:

Используя структуры данных, разработанные для лабораторной работы №6 (контейнер первого уровня и классы-фигуры) разработать алгоритм быстрой сортировки для класса-контейнера.

Необходимо разработать два вида алгоритма:

- Обычный, без параллельных вызовов.
- С использованием параллельных вызовов. В этом случае, каждый рекурсивный вызов сортировки должен создаваться в отдельном потоке.

Для создания потоков использовать механизмы:

- future
- packaged_task/async

Для обеспечения потоковой безопасности структур данных использовать:

- mutex
- lock_guard

Код:

ЛИСТИНГ ФАЙЛА TCONTAINER.H

```
# #ifndef TCONTAINER_H_INCLUDED

#define TCONTAINER_H_INCLUDED


#include <memory>

#include <iostream>

#include <exception>

#include <cstdlib>

#include <future>

#include "TIterator.h"

template <class T>

class TContainer
```

```
{
    private:
        std::shared_ptr<T>*  massive_Body;
        unsigned int massive_Size;
        unsigned int massive_Capacity ;
    public:
        TContainer(){
            this->massive_Capacity = 10;
            this->massive_Size = 0;
            this->massive_Body = new std::shared_ptr<T>[10];
        }
        TContainer(int t){
            this->massive_Capacity = t;
            this->massive_Size = t;
            this->massive_Body = new std::shared_ptr<T>[t];
        }
        bool Push_back (std::shared_ptr<T> &data){
            if (this->massive_Size == this->massive_Capacity){
                this->massive_Capacity = this->massive_Capacity *
1.5;
                std::shared_ptr<T> new_body [this-
>massive_Capacity];
                for (int i = 0; i < this->massive_Size; i++){
                    new_body[i] = this->massive_Body[i];
                    //free(this->massive_Body[i]);
                }
                this->massive_Body = new_body;
            }
            if (this->massive_Body == nullptr){
                free(this->massive_Body);
                this->massive_Capacity = 0;
                this->massive_Size = 0;
                return false;
```

```
            }
            this->massive_Body[this->massive_Size] = data;
            ++this->massive_Size;
            return true;
        }
        bool empty(){
            return this->Size() == 0;
        }
        std::shared_ptr<T> Pop_back(){
            std::shared_ptr<T> backtrack = this->massive_Body[this-
>massive_Size-1];
            this->massive_Body[this->massive_Size-1] = nullptr;
            this->massive_Size -= 1;
            return backtrack;
        }
        int Size() const {
            return this->massive_Size;
        }
        bool Set(int pos, std::shared_ptr<T> data){
            if(pos >= this->massive_Size){
                return false;
            }
            this->massive_Body[pos-1] = data;;
            return true;
        }
        std::shared_ptr<T> Get(int pos)const {
            return (this->massive_Body[pos]);
        }
        std::future<void> sort_in_background() {
            std::packaged_task<void(void) >
task(std::bind(std::mem_fn(&TContainer<T>::t_sorte), this));
            std::future<void> res(task.get_future());
            std::thread th(std::move(task));
```

```
            th.detach();
            return res;
        }
        void t_sorte(){
            if(massive_Size > 1){
            std::cout << "Sorting with merge_paral, size = " <<
massive_Size << '\n';
            TContainer* left = new TContainer<Figure>(), *right =
new TContainer<Figure>();
            std::shared_ptr<Figure> mid =
massive_Body[massive_Size/2];
            for (int i = 0; i < massive_Size; i++){
                    if (i != massive_Size/2){
                        if (massive_Body[i]->Square() > mid-
>Square()){
                            right->Push_back(massive_Body[i]);
                        }
                        else left->Push_back(massive_Body[i]);
                    }
            }
            std::cout << "filled\n";
            std::future<void> left_res = left->sort_in_background();
            std::future<void> right_res = right-
>sort_in_background();
            left_res.get();
            int i;
            for(i = 0; i < left->Size(); i++){
                massive_Body[i] = left->Get(i);
            }
            massive_Body[i] = mid;
            i++;
            right_res.get();
            for (int j = 0; j < right->Size(); i++){
                massive_Body[i+j] = right->Get(j);
```

```
            }
            std::cout << "copied\n";
            delete(left);
            delete(right);
            }
        return;
        }
        void sorte(){
            if (massive_Size > 1){
                std::cout << "Sorting with merge_non_paral, size = "
<< massive_Size << '\n';
                TContainer* left = new TContainer<Figure>(), *right
= new TContainer<Figure>();
                std::shared_ptr<Figure> mid =
massive_Body[massive_Size/2];


                for (int i = 0; i < massive_Size; i++){
                    if (i != massive_Size/2){
                        if (massive_Body[i]->Square() > mid-
>Square()){
                            right->Push_back(massive_Body[i]);
                        }
                        else left->Push_back(massive_Body[i]);
                    }
                }
                std::cout << "filled\n";
                left->sorte();
                right->sorte();


                int i;
                for(i = 0; i < left->Size(); i++){
                    massive_Body[i] = left->Get(i);
                }
                massive_Body[i] = mid;
```

```
                i++;
                for (int j = 0; j < right->Size(); i++){
                    massive_Body[i+j] = right->Get(j);
                }
                std::cout << "copied\n";
                delete(left);
                delete(right);
                return;
            }
        }
        friend std::ostream& operator << (std::ostream& os,const
TContainer<T>* massive){
            for (int i = 0; i < massive->Size(); i++){
                std::cout << i << "th element is:\n";
                std::shared_ptr<T> p = massive->Get(i);
                p->Print();
                std::cout << "\n\n";
            }
            return os;
        }
        TIterator <T> begin(){
            return TIterator<T>(massive_Body,0);
        }
        TIterator <T> end(){
            if (this->Size() > 0){
                return TIterator<T>(massive_Body,this->Size());
            }
            return TIterator<T>(nullptr, 0);
        }
        ~TContainer(){
            for (int i = 0; i < this->Size(); i++){
                this->massive_Body[i] = nullptr;
            }
```

```
        }
};
#endif // TContainer_H_INCLUDED
```

Выводы:

В ходе выполнения лабораторной работы был получен навык работы с потоками в C++. Я научился создавать потоки и работать с объектами в этих потоках. Были спроектированы и запрограммированы на языке C++ две сортировки динамических структур содержащих объекты классов фигур: быстрая сортировка и многопоточная быстрая сортировка.

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа №9

Дисциплина: «Объектно-ориентированное программирование»

Группа: 8О-206Б

Студент: Максимов А.Е.

Преподаватели:

Дзюба Д.В.

Поповкин А.В.

Москва, 2018

Лабораторная работа №9

Цели:

- Знакомство с лямбда-выражениями

Задание:

Используя структуры данных, разработанные для лабораторной работы №6 (контейнер первого уровня и классы-фигуры) необходимо разработать:

- Контейнер второго уровня с использованием шаблонов.
- Реализовать с помощью лямбда-выражений набор команд, совершающих операции над контейнером 1-го уровня:
    - Генерация фигур со случайным значением параметров;
    - Печать контейнера на экран;
    - Удаление элементов со значением площади меньше определенного числа;
- В контейнер второго уровня поместить цепочку команд.
- Реализовать цикл, который проходит по всем командам в контейнере второго уровня и выполняет их, применяя к контейнеру первого уровня.

Код:

ЛИСТИНГ ФАЙЛА TCONTAINER.H

```
# #include <cstdlib>
#include <algorithm>
#include "Triangle.h"
#include <thread>
#include "Pentagon.h"
#include "TContainer.h"
#include "Hexagon.h"
#include "Octagon.h"
#include "TIterator.h"
#include "MyStack.h"
int main() {
    char command;
```

```
    std::cin >> command;


    TContainer<Figure>* my_TContainer;
    std::shared_ptr<Figure> my_figure = nullptr;


    my_TContainer = new TContainer<Figure>();
    MyStack* my_stack = new MyStack();



    while (command != 'q'){


            if (command == 'a') {
                int pos;
                std::cout << "Please, type in the position of your
figure!" << std::endl;
                std::cin >> pos;
                if( pos >= my_TContainer->Size()) {
                        std::cout << "You have tried accessing a
nonexistent field! Quitting the program." << std::endl;
                        return 1;
                }


                std::cout << "Please type the first letter of the
type of your figure." << std::endl;
                std::cin >> command;
                if (command == 'p'){
                    std::cout << "You chose to add a pentagon!" <<
std::endl;


                    std::cout << "Please, type in five points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Pentagon(std::cin));
                }
```

```
                if (command == 'h'){

                    std::cout << "You chose to add a hexagon!" <<
std::endl;



                    std::cout << "Please, type in six points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;

                    my_figure = std::shared_ptr<Figure>(new
Hexagon(std::cin));

                }

                if (command == 'o'){

                    std::cout << "You chose to add a octagon!" <<
std::endl;



                    std::cout << "Please, type in eight points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;

                    my_figure = std::shared_ptr<Figure>(new
Octagon(std::cin));

                }



                my_TContainer->Set(pos, my_figure);

                //break;

            }



            else if (command == 'p'){

                std::cout << "Please type the first letter of the
type of your figure." << std::endl;

                std::cin >> command;

                if (command == 'p'){

                    std::cout << "You chose to add a pentagon!" <<
std::endl;



                    std::cout << "Please, type in five points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
```

```
                    my_figure = std::shared_ptr<Figure>(new
Pentagon(std::cin));
                }
                if (command == 'h'){
                    std::cout << "You chose to add a hexagon!" <<
std::endl;

                    std::cout << "Please, type in six points in std
coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Hexagon(std::cin));
                }
                if (command == 'o'){
                    std::cout << "You chose to add a octagon!" <<
std::endl;

                    std::cout << "Please, type in eight points in
std coordinate system. Do not forget the order!!" << std::endl <<
std::endl;
                    my_figure = std::shared_ptr<Figure>(new
Octagon(std::cin));
                }

                if(!(my_TContainer->Push_back(my_figure))){
                    std::cout << "Something went wrong! Quitting the
program." << std::endl;
                    return 1;
                }

            }

            else if (command == 's'){
                std::cout << "The size of your vector is " <<
my_TContainer->Size() << std::endl;
            }
```

```
            else if (command == 'o'){
                std::cout << my_TContainer;
            }

            else if (command == 't'){
                my_TContainer->t_sorte();
            }

            else if (command == 'm'){
                my_TContainer->sorte();
            }

            else if (command == 'd'){
                if (!(my_TContainer->empty())){
                    my_TContainer->Pop_back();
                    std::cout << "Deleted the last element from the
vector." << std::endl;
                }
                else {
                    std::cout << "Do not try this at home!\nTrying
to delete from an empty vector gone wrong!" << std::endl;
                }
            }
            else if (command == 'i'){
                std::cout << "Printing with iterators." <<
std::endl;
                for (auto i : *my_TContainer){
                    std::cout << "Printing element:" << std::endl;
                    i->Print();
                }
            }
            else if (command == '*'){
                std::cout << "Write the special operation: " ;
                std::cin >> command;
```

```
                my_stack->Push(command);
            }
            else if (command == '!'){
                while (my_stack->Empty()){
                    char t = my_stack->Pop();
                    if (t == 'r'){
                    std::for_each(my_TContainer-
>begin(),my_TContainer->end(),[](std::shared_ptr<Figure> _n){
                                _n = nullptr;
                                int var = rand()%3;
                                int modif = rand() % 25;
                                if(var == 0){
                                    _n = std::shared_ptr<Figure>(new
Pentagon(0+modif, 0+modif, 1+modif, 2+modif,
2+modif,0+modif,1+modif,2+modif,1+modif,0+modif));
                                }
                                else if (var == 1){
                                    _n = std::shared_ptr<Figure>(new
Hexagon(0+modif, 0+modif, 1+modif, 2+modif, 2+modif,
1+modif,0+modif,1+modif,2+modif,1+modif,0+modif,modif-1));
                                }
                                else if (var == 2){
                                    _n = std::shared_ptr<Figure>(new
Octagon(0+modif, 1+modif, 2+modif, 1+modif, 0+modif, modif-1, modif-
2, modif-1,2+modif,1+modif,modif,modif-1,modif-2,modif-
1,modif,1+modif));
                                }
                            });
                    }else if(t == 'p'){
                        std::for_each(my_TContainer-
>begin(),my_TContainer->end(),[](std::shared_ptr<Figure> _n){
                                    _n->Print();
                                    std::cout << "\n";
                                });
                    }else if (t == 'd'){
                        double sq;
```

```
                        std::cin >> sq;
                        std::for_each(my_TContainer-
>begin(),my_TContainer->end(),[](std::shared_ptr<Figure> _n){
                                        double sq;
                                        std::cin >> sq;
                                        if (_n->Square() < sq){
                                        _n =
std::shared_ptr<Figure>(new Pentagon());
                                    }
                                    });
                    }
                }
            }
            else {
                    std::cout << "Wrong command!" << std::endl;
            }
            my_figure = nullptr;
            std::cout << "Please, enter your next command: ";
            std::cin >> command;
    }
    delete my_TContainer;

    return 0;
}
```

Выводы:

В процессе выполнения данной лабораторной работы я познакомился с лямбда-выражениями. Лямбда-выражение используют для определения анонимного объекта-функции непосредственно в месте его вызова или передачи в функцию в качестве аргумента.