Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Основы метапрограммирования.

Студент:	Макаренкова В.М.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	13
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Pentagon.h:

```
#ifndef PENTAGON_H_
#define PENTAGON_H_
#include <iostream>
#include <cmath>
#include "point.h"



template<class T>
struct pentagon {
  Point<T> p[5];
  pentagon(std::istream& is);
  double area() const;
  Point<T> center() const;
  void print(std::ostream& os) const;
};



template<class T>
double pentagon<T>::area() const {
    const T d1 = triangle_height(p[0], p[1], p[2]) * sqrt((p[2].x - p[1].x) * (p[2].x
- p[1].x) + (p[2].y - p[1].y) * (p[2].y - p[1].y)) / 2;
    const T d2 = triangle_height(p[0], p[2], p[3]) * sqrt((p[3].x - p[2].x) * (p[3].x
- p[2].x) + (p[3].y - p[2].y) * (p[3].y - p[2].y)) / 2;
    const T d3 = triangle_height(p[0], p[3], p[4]) * sqrt((p[4].x - p[3].x) * (p[4].x
- p[3].x) + (p[4].y - p[3].y) * (p[4].y - p[3].y)) / 2;
    return d1 + d2 + d3;
}


template<class T>
Point<T> pentagon<T>::center() const {
  Point<T> res;
  res.x = (p[0].x + p[1].x + p[2].x + p[3].x + p[4].x) / 5;
  res.y = (p[0].y + p[1].y + p[2].y + p[3].y + p[4].y) / 5;
  return res;
}
```

```
template<class T>
void pentagon<T>::print(std::ostream& os) const{
     for(int i = 0; i < 5; ++i){
      os << p[i];
    if(i + 1 != 5){
      os << ' ';
    }
  }
}


template<class T>
pentagon<T>::pentagon(std::istream& is) {
     for(int i = 0; i < 5; ++i){
    is >> p[i];
  }
}

#endif
```

Point.h:

```
#ifndef Point_H_
#define Point_H_

#include <iostream>
#include <cmath>
#include <limits>

template<class T>
struct Point {
  T x;
  T y;
};

template<class T>
Point<T> operator+(const Point<T>& A, const Point<T>& B) {
    Point<T> res;
    res.x = A.x + B.x;
    res.y = A.y + B.y;
    return res;
}
```

```
template<class T>
Point<T> operator/=(Point<T>& A, const double B) { //деление на число
    A.x /= B;
    A.y /= B;
    return A;
}

template<class T>
std::istream& operator>> (std::istream& is, Point<T>& p) {
  is >> p.x >> p.y;
  return is;
}

template<class T>
std::ostream& operator<< (std::ostream& os, const Point<T>& p) {
  os << '(' << p.x << ';' << p.y << ')' ;
  return os;
}

template <class T>
double triangle_height(Point<T> p1, Point<T> p2, Point<T> p3) {
    double A = p2.y - p3.y;
    double B = p3.x - p2.x;
    double C = p2.x*p3.y - p3.x*p2.y;
    return (std::abs(A*p1.x + B*p1.y + C) / std::sqrt(A*A + B*B));
}


#endif
```

Hexagon.h:

```
#ifndef D_HEXAGON_H
#define D_HEXAGON_H

#include <iostream>
#include <cmath>
#include "point.h"



template<class T>
struct hexagon {
  Point<T> p[6];
  hexagon(std::istream& is);
```

```
  double area() const;
  Point<T> center() const;
  void print(std::ostream& os) const;
};



template<class T>
double hexagon<T>::area() const {
    return (-0.5) * ((p[0].x*p[1].y + p[1].x*p[2].y + p[2].x*p[3].y + p[3].x*p[4].y +
p[4].x*p[5].y + p[5].x*p[0].y) - ( p[0].y*p[1].x + p[1].y*p[2].x + p[2].y*p[3].x +
p[3].y*p[4].x + p[4].y*p[5].x + p[5].y*p[0].x ));

}



template<class T>
Point<T> hexagon<T>::center() const {
      Point<T> res;
  res.x = (p[0].x + p[1].x + p[2].x + p[3].x + p[4].x + p[5].x) / 6;
  res.y = (p[0].y + p[1].y + p[2].y + p[3].y + p[4].y + p[5].y) / 6;
  return res;
}



template<class T>
void hexagon<T>::print(std::ostream& os) const{
     for(int i = 0; i < 6; ++i){
      os << p[i];
    if(i + 1 != 6){
      os << ' ';
    }
  }
}



template<class T>
hexagon<T>::hexagon(std::istream& is) {
     for(int i = 0; i < 6; ++i){
    is >> p[i];
  }
```

```
}

#endif
```
Rhombus.h:
```
#ifndef RHOMBUS_H_
#define RHOMBUS_H_
#include <iostream>
#include <cmath>
#include "point.h"


template<class T>
struct rhombus {
  Point<T> p[4];
  rhombus(std::istream& is);
  double area() const;
  Point<T> center() const;
  void print(std::ostream& os) const;
};

template<class T>
rhombus<T>::rhombus(std::istream& is) {
  for(int i = 0; i < 4; ++i){
    is >> p[i];
  }
  double a, b, c, d;
  a = sqrt((p[1].x - p[0].x) * (p[1].x - p[0].x) + (p[1].y - p[0].y) * (p[1].y - p[0].y));
  b = sqrt((p[2].x - p[1].x) * (p[2].x - p[1].x) + (p[2].y - p[1].y) * (p[2].y - p[1].y));
  c = sqrt((p[2].x - p[3].x) * (p[2].x - p[3].x) + (p[2].y - p[3].y) * (p[2].y - p[3].y));
  d = sqrt((p[3].x - p[0].x) * (p[3].x - p[0].x) + (p[3].y - p[0].y) * (p[3].y - p[0].y));
  if(a != b || a != c || a != d)
      throw std::logic_error("Wrong coordinates. It's not a rhombus.");
}

template<class T>
double rhombus<T>::area() const {
  const T d1 =  sqrt((p[0].x - p[2].x) * (p[0].x - p[2].x) + (p[0].y - p[2].y) * (p[0].y - p[2].y));
```

```
  const T d2 =  sqrt((p[1].x - p[3].x) * (p[1].x - p[3].x) + (p[1].y - p[3].y) *
(p[1].y - p[3].y));
  return d1 * d2 / 2;
}

template<class T>
Point<T> rhombus<T>::center() const {
  Point<T> res;
  res.x = (p[0].x + p[1].x + p[2].x + p[3].x) / 4;
  res.y = (p[0].y + p[1].y + p[2].y + p[3].y) / 4;
  return res;
}

template<class T>
void rhombus<T>::print(std::ostream& os) const {
  for(int i = 0; i < 4; ++i){
      os << p[i];
    if(i + 1 != 4){
      os << ' ';
    }
  }
}

#endif
```

Templates.h:

```
#ifndef TEMPLATES_H_
#define TEMPLATES_H_

#include <tuple>
#include <type_traits>
#include "point.h"


template<class T>
struct is_point : std::false_type {};

template<class T>
struct is_point<Point<T>> : std::true_type {};

template<class T>
struct is_figurelike_tuple : std::false_type {};

template<class Head, class... Tail>
```

```
struct is_figurelike_tuple<std::tuple<Head, Tail...>> :
  std::conjunction<is_point<Head>, std::is_same<Head, Tail>...> {};

template<class T>
inline constexpr bool is_figurelike_tuple_v = is_figurelike_tuple<T>::value;



template<class T, class = void>
struct has_method_area : std::false_type {};

template<class T>
struct has_method_area<T, std::void_t<decltype(std::declval<const T&>().area())>> : std::true_type {};

template<class T>
inline constexpr bool has_method_area_v = has_method_area<T>::value;

template<class T>
std::enable_if_t<has_method_area_v<T>, double> area(const T& object) {
  return object.area();
}



template<class T, class = void>
struct has_method_center : std::false_type {};

template<class T>
struct has_method_center<T, std::void_t<decltype(std::declval<const T&>().center())>> : std::true_type {};

template<class T>
inline constexpr bool has_method_center_v = has_method_center<T>::value;

template<class T>
std::enable_if_t<has_method_center_v<T>, Point<double>> center(const T& object) {
    return object.center();
}
```

```
template<class T, class = void>
struct has_method_print : std::false_type {};

template<class T>
struct has_method_print<T, std::void_t<decltype(std::declval<const T&>().print(std::cout))>> : std::true_type {};

template<class T>
inline constexpr bool has_method_print_v = has_method_print<T>::value;

template<class T>
std::enable_if_t<has_method_print_v<T>, void> print(std::ostream& os, const T& object) {
    object.print(os);
}


template<size_t Id, class T>
  double compute_area(const T& tuple) {
  if constexpr (Id >= std::tuple_size_v<T>){
      return 0;
    }else{
    const auto x1 = std::get<Id - 0>(tuple).x - std::get<0>(tuple).x;
    const auto y1 = std::get<Id - 0>(tuple).y - std::get<0>(tuple).y;
    const auto x2 = std::get<Id - 1>(tuple).x - std::get<0>(tuple).x;
    const auto y2 = std::get<Id - 1>(tuple).y - std::get<0>(tuple).y;
    const double local_area = std::abs(x1 * y2 - y1 * x2) * 0.5;
    return local_area + compute_area<Id + 1>(tuple);
  }
}

template<class T>
std::enable_if_t<is_figurelike_tuple_v<T>, double>
area(const T& object) {
  if constexpr (std::tuple_size_v<T> < 3){
      throw std::logic_error("It`s not a figure");
  }else{
    return compute_area<2>(object);
  }
}

template<size_t Id, class T>
Point<double> tuple_center(const T& object) {
```

```
    if constexpr (Id >= std::tuple_size<T>::value) {
        return Point<double> {0, 0};
    } else {
        Point<double> res = std::get<Id>(object);
        return res + tuple_center<Id+1>(object);
    }
}

template<class T>
Point<double> compute_center(const T &tuple) {
    Point<double> res{0, 0};
    res = tuple_center<0>(tuple);
    res /= std::tuple_size_v<T>;
    return res;
}

template<class T>
std::enable_if_t<is_figurelike_tuple_v<T>, Point<double>>
center(const T& object) {
    if constexpr (std::tuple_size_v<T> < 3){
        throw std::logic_error("It`s not a figure");
    }else{
        return compute_center(object);
    }
}

template<size_t Id, class T>
void step_print(const T& object, std::ostream& os) {
    if constexpr (Id >= std::tuple_size<T>::value) {
        std::cout << "\n";
    } else {
        os << std::get<Id>(object) << " ";
        step_print<Id + 1>(object, os);
    }
}

template<class T>
std::enable_if_t<is_figurelike_tuple_v<T>, void>
print(std::ostream& os, const T& object) {
    if constexpr (std::tuple_size_v<T> < 3){
        throw std::logic_error("It`s not a figure");
    }else{
        step_print<0>(object, os);
```

```
    }
}
#endif
```
Main.cpp:
```
#include <iostream>
#include <tuple>
#include "point.h"
#include "hexagon.h"
#include "pentagon.h"
#include "rhombus.h"
#include "templates.h"

template<class T>
void init(std::istream& is, std::ostream& os) {
    if constexpr (is_figurelike_tuple<T>::value) {
        int arg;
        std::cin >> arg;
        std::cout << "Input coordinates: " << std::endl;
        if (arg == 4) {
            Point<double> A, B, C, D;
            is >> A >> B >> C >> D;
            auto object = std::make_tuple(A, B, C, D);
            print(os, object);
            os << "Area: " << area(object) << std::endl;
            os << "Center: " << center(object) << std::endl;
        }else if(arg == 5){
            Point<double> A, B, C, D, F;
            is >> A >> B >> C >> D >> F;
            auto object = std::make_tuple(A, B, C, D, F);
            print(os, object);
            os << "Area: " << area(object) << std::endl;
            os << "Center: " << center(object) << std::endl;
        }else if(arg == 6){
            Point<double> A, B, C, D, F, G;
            is >> A >> B >> C >> D >> F >> G;
            auto object = std::make_tuple(A, B, C, D, F, G);
            print(os, object);
            os << "Area: " << area(object) << std::endl;
            os << "Center: " << center(object) << std::endl;
        }
        }else {
        T object(is);
        print(os, object);
```

```
        os << '\n' <<"Area: " << area(object) << std::endl;
        os << "Center: " << center(object) << std::endl;
    }
}


int main() {
    char obj_type;
    std:: cout<< " Available input:\n1 - input pentagon\n2 - input Rhombus\n3
- input hexagon\n4 - Tuple\n5 - Exit" <<std::endl;
    while (std::cin >> obj_type){
        if(obj_type == '4') {
            std::cout << "Input number of vertices: " << std::endl;
            init<std::tuple<Point<double>>> (std::cin, std::cout);
        }else if(obj_type == '1'){
            std::cout << "Input pentagon coordinates: " << std::endl;
            init<pentagon<double>>(std::cin, std::cout);
        }else if(obj_type == '2'){
            std::cout << "Input Rhombus coordinates: " << std::endl;
            init<rhombus<double>>(std::cin, std::cout);
        }else if(obj_type == '3'){
            std::cout << "Input hexagon coordinates: " << std::endl;
            init<hexagon<double>>(std::cin, std::cout);
        }else if(obj_type == '5'){
             return 0;
        }else{
             std::cout << "Try another key" << std::endl;
        }
    }
}
```

2. Ссылка на репозиторий на GitHub.

https://github.com/vera0000/OOP2019

3. Набор тестов.

test1.test:

```
2
0 0 0 2 2 2 2 0
1
1 1 0 2 1 3 2 3 3 2
3
```

2 0 1 2 2 4 4 4 5 2 4 0
test2.test:
2
0 0 1 1 2 2 3 3

4. Результаты выполнения тестов.

test1.result:
Available input:
1 - input pentagon
2 - input Rhombus
3 - input hexagon
4 - Tuple
5 - Exit
Input Rhombus coordinates:
(0;0) (0;2) (2;2) (2;0)
Area: 4
Center: (1;1)
Input pentagon coordinates:
(1;1) (0;2) (1;3) (2;3) (3;2)
Area: 3.5
Center: (1.4;2.2)
Input hexagon coordinates:
(2;0) (1;2) (2;4) (4;4) (5;2) (4;0)
Area: 12
Center: (3;2)

test.result:
Available input:
1 - input pentagon
2 - input Rhombus
3 - input hexagon
4 - Tuple
5 - Exit
2
Input Rhombus coordinates:
0 0 1 1 2 2 3 3
terminate called after throwing an instance of 'std::logic_error'
  what(): Wrong coordinates. It's not a rhombus.
Aborted (core dumped)

5. Объяснение результатов работы программы.

1) Шаблонная функция center() возвращает точку с x –деление суммы иксов всех точек данной фигуры на их количество, y – аналогично x. Она определена для моих фигур и tuple.
2) Функция print() печатает координаты всех точек данной фигуры или кортежа.
Она определена для моих фигур и tuple.

3) Функция area() вычисляет площадь данной фигуры или совокупности точек и возвращает это значение.

6. Вывод.

Выполняя данную лабораторную я получила опыт работы с шаблонами в C++. Узнал о применении шаблонов в метапрограммировании. Также я познакомилась с полезными заголовочными файлами <tuple> и <type_traits>, освоил enable_if, decltype и базовые вещи для работы с tuple.
Данная лабораторная работа показала многогранность и мощь языка C++.

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Итераторы и умные указатели.

Студент:	Макаренкова В.М.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Main.cpp

```
#include <iostream>
#include <algorithm>

#include "list.h"
#include "point.h"
#include "rhombus.h"

#include<clocale>


void Menu1() {
    std::cout << "1. Добавить фигуру в список\n";
    std::cout << "2. Удалить фигуру\n";
    std::cout << "3. Вывести фигуру\n";
    std::cout << "4. Вывести все фигуры\n";
    std::cout << "5. Вывести кол-во фигур чья площадь больше чем ...\n";
}

void PushMenu() {
    std::cout << "1. Добавить фигуру в начало списка\n";
    std::cout << "2. Добавить фигуру в конец списка\n";
    std::cout << "3. Добавить фигуру по индексу\n";
}

void DeleteMenu() {
    std::cout << "1. Удалить фигуру в начале списка\n";
    std::cout << "2. Удалить фигуру в конце списка\n";
    std::cout << "3. Удалить фигуру по индексу\n";
}

void PrintMenu() {
    std::cout << "1. Вывести первую фигуру в списке\n";
    std::cout << "2. Вывести последнюю фигуру в списке\n";
    std::cout << "3. Вывести фигуру по индексу\n";
}

int main() {
    setlocale(LC_ALL, "rus");
    containers::list<rhombus<int>> MyList;
```

```
rhombus<int> Temprhombus;

while (true) {
    Menu1();
    int n, m, kek;
    double s;
    std::cin >> n;
    switch (n) {
    case 1:
        Temprhombus.read(std::cin);
        PushMenu();
        std::cin >> m;
        switch (m) {
        case 1:
            MyList.push_front(Temprhombus);
            break;
        case 2:
            MyList.push_back(Temprhombus);
            break;
        case 3:
            std::cin >> kek;
            MyList.insert_by_number(kek, Temprhombus);
        default:
            break;
        }
        break;
    case 2:
        DeleteMenu();
        std::cin >> m;
        switch (m) {
        case 1:
            MyList.pop_front();
            break;
        case 2:
            MyList.pop_back();
            break;
        case 3:
            std::cin >> kek;
            MyList.delete_by_number(kek);
            break;
        default:
            break;
```

```
            }
            break;
        case 3:
            PrintMenu();
            std::cin >> m;
            switch (m) {
            case 1:
                MyList.front().print(std::cout);
                std::cout << std::endl;
                break;
            case 2:
                MyList.back().print(std::cout);
                std::cout << std::endl;
                break;
            case 3:
                std::cin >> kek;
                MyList[kek].print(std::cout);
                std::cout << std::endl;
                break;
            default:
                break;
            }
            break;
        case 4:
            std::cout << MyList.length() << std::endl;
            std::for_each(MyList.begin(), MyList.end(), [](rhombus<int>& X) {
X.print(std::cout); std::cout << std::endl; });

            break;
        case 5:
            std::cin >> s;
            std::cout      <<      std::count_if(MyList.begin(),      MyList.end(),
[=](rhombus<int>& X) {return X.area() > s; }) << std::endl;
            break;
        default:
            return 0;
        }
    }

    system("pause");
    return 0;
}
```

2. Набор тестов.

test1.test:

```
1
0 0 0 2 2 2 2 0
1
2
1
2 0 1 2 2 4 3 2
1
4
5
1
```

test2.test:

```
1
0 0 1 1 2 2 3 3
```

4. Результаты выполнения тестов.

test1.result:

```
  1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
1
0 0 0 2 2 2 2 0
1. Добавить фигуру в начало списка
2. Добавить фигуру в конец списка
3. Добавить фигуру по индексу
2
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
1
2 0 1 2 2 2 4 3 2
1. Добавить фигуру в начало списка
2. Добавить фигуру в конец списка
3. Добавить фигуру по индексу
1
1. Добавить фигуру в список
```

2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
4
2
(2;0) (1;2) (2;4) (3;2)
(0;0) (0;2) (2;2) (2;0)
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
5 1
2
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
test2.result:
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
1
0 0 1 1 2 2 3 3
terminate called after throwing an instance of 'std::logic_error'
  what(): Wrong coordinates. It's not a rhombus.
Aborted

6. Вывод.

Выполняя данную лабораторную, получила опыт в работе с итераторами и умными указателями в C++. Узнала о применении итераторов в STL. Было трудно работать с умными указателями, но, если научиться грамотно использовать их, можно избежать неприятных утечек памяти.

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Аллокатор.

Студент:	Макаренкова В.М.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Main.cpp

```
#include<iostream>
#include<algorithm>
#include<locale.h>
#include"rhombus.h"
#include"list.h"
#include"allocator.h"

void Menu1() {
 std::cout << "1. Добавить фигуру в список\n";
 std::cout << "2. Удалить фигуру\n";
 std::cout << "3. Вывести фигуру\n";
 std::cout << "4. Вывести все фигуры\n";
 std::cout << "5. Вывести кол-во фигур чья площаль больше чем ...\n";
}

void PushMenu() {
 std::cout << "1. Добавить фигуру в начало списка\n";
 std::cout << "2. Добавить фигуру в конец списка\n";
 std::cout << "3. Добавить фигуру по индексу\n";
}

void DeleteMenu() {
 std::cout << "1. Удалить фигуру в начале списка\n";
 std::cout << "2. Удалить фигуру в конце списка\n";
 std::cout << "3. Удалить фигуру по индексу\n";
}

void PrintMenu() {
 std::cout << "1. Вывести первую фигуру в списке\n";
 std::cout << "2. Вывести последнюю фигуру в списке\n";
 std::cout << "3. Вывести фигуру по индексу\n";
}

int main() {
 std::cout<<sizeof(rhombus<int>)<<std::endl;
 setlocale(LC_ALL, "rus");
 containers::list<rhombus<int>,  allocators::my_allocator<rhombus<int>, 500>> MyList;

 rhombus<int> Tmprhombus;

 while (true) {
```

```
Menu1();
int n, m, ind;
double s;
std::cin >> n;
switch (n) {
case 1:
    Tmprhombus.read(std::cin);
    PushMenu();
    std::cin >> m;
    switch (m) {
    case 1:
        MyList.push_front(Tmprhombus);
        break;
    case 2:
        MyList.push_back(Tmprhombus);
        break;
    case 3:
        std::cin >> ind;
        MyList.insert_by_number(ind, Tmprhombus);
    default:
        break;
    }
    break;
case 2:
    DeleteMenu();
    std::cin >> m;
    switch (m) {
    case 1:
        MyList.pop_front();
        break;
    case 2:
        MyList.pop_back();
        break;
    case 3:
        std::cin >> ind;
        MyList.delete_by_number(ind);
        break;
    default:
        break;
    }
    break;
case 3:
    PrintMenu();
    std::cin >> m;
    switch (m) {
```

```
                case 1:
                    MyList.front().print(std::cout);
                    std::cout << std::endl;
                    break;
                case 2:
                    MyList.back().print(std::cout);
                    std::cout  << std::endl;
                    break;
                case 3:
                    std::cin >> ind;
                    MyList[ind].print(std::cout);
                    std::cout  << std::endl;
                    break;
                default:
                    break;
                }
                break;
            case 4:
                std::for_each(MyList.begin(), MyList.end(), [](rhombus<int> &X)
        { X.print(std::cout); std::cout << std::endl; });

                break;
            case 5:
                std::cin >> s;
                std::cout      <<      std::count_if(MyList.begin(),      MyList.end(),
        [=](rhombus<int>& X) {return X.area() > s; }) << std::endl;
                break;
            default:
                return 0;
            }
         }

          system("pause");
          return 0;
        }
```

3. Набор тестов.

test1.test:

```
1
0 0 0 2 2 2 2 0
1
2
1
2 0 1 2 2 4 3 2
1
```

4
5
1
test2.test:
1
0 0 1 1 2 2 3 3

4. Результаты выполнения тестов.

test1.result:
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
1
0 0 0 2 2 2 2 0
1. Добавить фигуру в начало списка
2. Добавить фигуру в конец списка
3. Добавить фигуру по индексу
2
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
1
2 0 1 2 2 2 4 3 2
1. Добавить фигуру в начало списка
2. Добавить фигуру в конец списка
3. Добавить фигуру по индексу
1
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
4
2
(2;0) (1;2) (2;4) (3;2)
(0;0) (0;2) (2;2) (2;0)
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
5 1

```
2
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...

test2.result:
1. Добавить фигуру в список
2. Удалить фигуру
3. Вывести фигуру
4. Вывести все фигуры
5. Вывести кол-во фигур чья площадь больше чем ...
1
0 0 1 1 2 2 3 3
terminate called after throwing an instance of 'std::logic_error'
  what():  Wrong coordinates. It's not a rhombus.
Aborted
```

6. Вывод.

Выполняя данную лабораторную, я получила опыт работы с аллокаторами и умными указателями в C++. Узнала о применении аллокаторов в STL. Поняла, для чего нужны аллокаторы.

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Проектирование структуры классов.

Студент:	Макаренкова В.М.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	13
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

main.cpp

```
#include <iostream>
#include "factory.h"
#include "editor.h"

void crt(editor& edit) {
    std::string tmp;
    std::cin >> tmp;
    edit.crt_doc(tmp);
}

void load(editor& edit) {
    std::string tmp;
    std::cin >> tmp;
    try {
        edit.load_doc(tmp);
    } catch (std::runtime_error& e) {
        std::cout << e.what();
    }
}

void save(editor& edit) {
    std::string tmp;
    try {
        edit.save_doc();
    } catch (std::runtime_error& e) {
        std::cout << e.what();
    }
}

void add(editor& edit) {
    factory fac;
    try {
        std::shared_ptr<figure> newElem = fac.new_figure(std::cin);
        edit.add_doc(newElem);
    } catch (std::logic_error& e) {
        std::cout << e.what() << '\n';
    }
}

void rmv(editor& edit) {
    int index;
```

```
    std::cin >> index;
    try {
        edit.delete_doc(index);
    } catch (std::logic_error& err) {
        std::cout << err.what() << "\n";
    }
}

int main() {
    editor edit;
    std::string command;
    while (true) {
        std::cin >> command;
        if (command == "crt") {
            crt(edit);
        } else if (command == "load") {
            load(edit);
        } else if (command == "save") {
            save(edit);
        } else if (command == "ext") {
            break;
        } else if (command == "add") {
            add(edit);
        } else if (command == "rmv") {
            rmv(edit);
        } else if (command == "prt") {
            edit.prt_doc();
        } else if (command == "undo") {
            try {
                edit.undo();
            } catch (std::logic_error& e) {
                std::cout << e.what();
            }
        } else {
            std::cout << "no such a command\n";
        }
    }
    return 0;
}
```

3. Набор тестов

```
add
r
0 0 0 1 1 1 1 0
rmv
```

undo
crt t.txt
add r 0 0 0 1 1 1 1 0
prt
save
undo
prt

4. Результаты выполнения тестов.

no doc
no doc
empty history

rhombus
0 0

0 1

1 1

1 0

Buffer is empty

5. **Объяснение результатов работы программы.**

1) При запуске программы вводится одна из 8 возможных команд в виде строки.
2) crt – создание файла.
3) ext – корректный выход их программы.
4) load – загрузка файла.
5) save – сохранение файла.
6) add – добавление фигуры.
7) rmv – удаление фигуры.
8) prt – печать буфера.
9) undo – отмена операции.

5. Вывод.

Выполняя данную лабораторную, я получил опыт проектирования структуры классов с использованием системы сборки scons

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Асинхронное программирование.

Студент:	Макаренкова В.М.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	13
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Main.cpp

```
#include <iostream>
#include <iostream>
#include <memory>
#include <vector>
#include "factory.h"
#include "pentagon.h"
#include "rhombus.h"
#include "hexagon.h"
#include "subscriber.h"

int main(int argc,char* argv[]) {
    if (argc != 2) {
        std::cout << " Usage:\n" << argv[0] << " not enough arguments\n";
        return 1;
}
    const int32_t buffer_size = std::stoi(argv[1]);
    std::shared_ptr<std::vector<std::shared_ptr<figure>>> buffer =
            std::make_shared<std::vector<std::shared_ptr<figure>>>();
    buffer->reserve(buffer_size);
    factory factory;
    std::string command;
    std::cout << "add - add a new figure\n" << "exit - the end of the program\n";
    //thread
    subscriber sub;
    sub.processors.push_back(std::make_shared<stream_processor>());
    sub.processors.push_back(std::make_shared<file_processor>());
    std::thread sub_thread(std::ref(sub));

    while (true) {
        std::unique_lock<std::mutex> guard(sub.mtx);
        std::cout << "begin\n";
        std::cin >> command;
        if (command == "add") {
            try {
                buffer->push_back(factory.FigureCreate(std::cin));
            } catch (std::logic_error &e) {
                std::cout << e.what() << '\n';
                continue;
            }
            if (buffer->size() == buffer_size) {
```

```
            sub.buffer = buffer;
            sub.cv.notify_all();
            sub.cv.wait(guard, [&](){ return sub.buffer == nullptr;});
            buffer->clear();
        }
    } else if (command == "exit")  {

        break;
    } else {
        std::cout << "unknown command\n";
    }
  }
  sub.end = true;
  sub.cv.notify_all();
  sub_thread.join();


  return 0;
}
```

3. Набор тестов.

test1.test:

```
add pentagon 1 1 1 1 1 1 1 1 1 1
add pentagon 2 2 2 2 2 2 2 2 2 2
add pentagon 3 3 3 3 3 3 3 3 3 3
add pentagon 5 5 5 5 5 5 5 5 5 5
add hexagon 6 6 6 6 6 6 6 6 6 6 6 6
exit
```

test_02.test:

```
remove
add kek
end
exit
```

4. Результаты выполнения тестов.

test1.result:

```
add - adding a new shape
exit - the end of the program
begin
begin
begin
pentagon
1 1
1 1
```

1 1
1 1
1 1
pentagon
2 2
2 2
2 2
2 2
2 2
pentagon
3 3
3 3
3 3
3 3
3 3
begin
begin
begin
pentagon
5 5
5 5
5 5
5 5
5 5
hexagon
6 6
6 6
6 6
6 6
6 6
6 6
begin

test2.result:
add - adding a new shape
exit - the end of the program
begin
unknown command
begin
There is no such figure

begin
unknown command
begin

5. Объяснение результатов работы программы.

Работают 2 потока, один считывает и сохраняет в буфер фигуры, а другой печатает их в файл и на консоль.

6. Вывод.

Выполняя данную лабораторную работу, я обрела навыки многопоточного программирования, научилась использовать мьютексы и условные переменные.