

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Отчёт по Лабораторной работе №3
“Наследование, Полиморфизм“
по курсу “Объектно-Объективное Программирование“
III Семестр

Студент:	Катермин В.С.
Группа:	М8О-208Б-18
Преподаватель:	Журавлёв А.А.
Оценка:	
Дата:	14.10.19

1. **Тема:** Наследование, Полиморфизм в C++.

2. **Код программы:**

point.h

```
#ifndef D_POINT_H
#define D_POINT_H

#include <iostream>

struct point{
    double x,y;
};

std::ostream& operator << (std::ostream& os, const point& p);

#endif //D_POINT_H
```

point.cpp

```
#include "point.h"

std::ostream& operator << (std::ostream& os, const point& p){
    return os << p.x << " " << p.y;
}
```

figure.h

```
#ifndef D_FIGURE_H
#define D_FIGURE_H

#include <iostream>

#include "point.h"

struct figure{
    virtual point center() = 0;
    virtual double area() = 0;
    virtual void print(std::ostream& os) = 0;

    virtual ~figure() {};

//std::ostream& operator << (std::ostream& os, const figure& f);
};

#endif //D_FIGURE_H
```

square.h

```
#ifndef D_SQUARE_H
#define D_SQUARE_H

#include "figure.h"

struct square : figure{
    square(const point& p, double length);
    square(std::istream& is);

    point center() override;
    double area() override;
    void print(std::ostream& os) override;

private:
    point p_;
    double length_;
};
```

```
};
```

```
#endif //D_SQUARE_H
```

square.cpp

```
#include "square.h"
```

```
#include <cmath>
```

```
//#include <algorithm>
```

```
square::square(const point& p, double length):  
    p_(p), length_(length) {}
```

```
square::square(std::istream& is) {  
    is >> p_.x >> p_.y >> length_;  
}
```

```
point square::center() {  
    return {(p_.x + (length_ * 0.5)), (p_.y + (length_ * 0.5))};  
}
```

```
double square::area() {  
    return length_ * length_;  
}
```

```
void square::print(std::ostream& os){  
    os << "Square [" << p_ << "]" (" << length_ << ");  
}
```

rectangle.h

```
#ifndef D_RECTANGLE_H
```

```
#define D_RECTANGLE_H
```

```
#include "figure.h"
```

```
struct rectangle : figure{  
    rectangle(const point& p1, const point& p2);  
    rectangle(std::istream& is);
```

```
    point center() override;  
    double area() override;  
    void print(std::ostream& os) override;
```

```
private:  
    point p1_, p2_;  
};
```

```
#endif //D_RECTANGLE_H
```

rectangle.cpp

```
#include "rectangle.h"
```

```
#include <cmath>
```

```
//#include <algorithm>
```

```
rectangle::rectangle(const point& p1, const point& p2):  
    p1_(p1), p2_(p2) {}
```

```
rectangle::rectangle(std::istream& is){  
    is >> p1_.x >> p1_.y >> p2_.x >> p2_.y;  
}
```

```
point rectangle::center() {
```

```

    return {(p1_.x + p2_.x) * 0.5, (p1_.y + p2_.y) * 0.5};
}

double rectangle::area() {
    const double dx = p1_.x - p2_.x;
    const double dy = p1_.y - p2_.y;
    return std::abs(dx * dy);
}

void rectangle::print(std::ostream& os) {
    os << "Rectangle [" << p1_ << "] [" << p2_ << "]\n";
}

```

trapeze.h

```

#ifndef D_TRAPEZE_H
#define D_TRAPEZE_H

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

struct trapeze : figure{
    trapeze(const point& p, double l1, double l2, double h);
    trapeze(std::istream& is);

    point center() override;
    double area() override;
    void print(std::ostream& os) override;

private:
    point p_;
    double l1_, l2_, h_;
};

#endif //D_TRAPEZE_H

```

trapeze.cpp

```

#include "trapeze.h"
#include <cmath>
// #include <algorithm>

trapeze::trapeze(const point& p, double l1, double l2, double h):
    p_(p), l1_(l1), l2_(l2), h_(h) {}

trapeze::trapeze(std::istream& is){
    is >> p_.x >> p_.y >> l1_ >> l2_ >> h_;
}

point trapeze::center() {
    return {(p_.x + (l1_ * 0.5)), (p_.y + (h_ * 0.5))};
}

double trapeze::area() {
    return std::abs((l1_+l2_)*h_*0.5);
}

void trapeze::print(std::ostream& os) {
    os << "Trapeze [" << p_ << "] (" << l1_ << " " << l2_ << " " << h_ << ")\n";
}

```

lab3.cpp

```

#include <iostream>

```

```

#include <vector>

#include "square.h"
#include "rectangle.h"
#include "trapeze.h"

int main(){
    std::vector<figure*> figures;
    for (;;) {
        int command;
        std::cin >> command;
        if (command == 0) {
            break;
        } else if (command == 1) {
            int figure_type;
            std::cin >> figure_type;
            figure* ptr;
            if (figure_type == 0) {
                ptr = new square(std::cin);
            } else if (figure_type == 1) {
                ptr = new rectangle(std::cin);
            } else {
                ptr = new trapeze(std::cin);
            }
            figures.push_back(ptr);
        } else if (command == 2) {
            int id;
            std::cin >> id;
            delete figures[id];
            figures.erase(figures.begin() + id);
        } else if (command == 3) {
            std::cout << "Centers:\n";
            for (figure* ptr: figures) {
                std::cout << ptr->center() << std::endl;
            }
        } else if (command == 4) {
            std::cout << "Areas:\n";
            for (figure* ptr: figures) {
                std::cout << ptr->area() << std::endl;
            }
        } else if (command == 5) {
            std::cout << "Figures:\n";
            for (figure* ptr: figures) {
                ptr->print(std::cout);
                std::cout << std::endl;
            }
        }
    }
    for (figure* ptr: figures) {
        delete ptr;
    }
}

```

Makefile

```
project(lab3)
```

```
add_executable(lab3
```

```
./lab3.cpp
```

```
./point.cpp
```

```
./square.cpp
```

```
./rectangle.cpp
```

```
./trapeze.cpp)
```

```
set(CMAKE_CXX_FLAGS
"${CMAKE_CXX_FLAGS} -Wall -Wextra")
```

3. Ссылка на репозиторий:

https://github.com/GitGood2000/oop_exercise_03

4. Набор testcases:

test_00.test

```
1
0
1 3 5
1
1
2 5 6 9
1
2
4 7 10 6 8
3
4
5
0
```

test_00.result

```
Centers:
3.5 5.5
4 7
9 11
Areas:
25
16
64
Figures:
Square [1 3] (5)
Rectangle [2 5] [6 9]
Trapeze [4 7] (10 6 8)
```

test_01.test

```
1
0
5 7 3
1
0
3 4 7.5
1
1
2 4 8 6
1
1
-2 -4 6 3
1
2
3 6 7 4 5.5
1
2
2 6 12 8 10
3
4
5
2
1
5
```

2
2
5
2
3
5
0

test_01.result

Centers:

6.5 8.5
6.75 7.75
5 5
2 -0.5
6.5 8.75
8 11

Areas:

9
56.25
12
56
30.25
100

Figures:

Square [5 7] (3)
Square [3 4] (7.5)
Rectangle [2 4] [8 6]
Rectangle [-2 -4] [6 3]
Trapeze [3 6] (7 4 5.5)
Trapeze [2 6] (12 8 10)

Figures:

Square [5 7] (3)
Rectangle [2 4] [8 6]
Rectangle [-2 -4] [6 3]
Trapeze [3 6] (7 4 5.5)
Trapeze [2 6] (12 8 10)

Figures:

Square [5 7] (3)
Rectangle [2 4] [8 6]
Trapeze [3 6] (7 4 5.5)
Trapeze [2 6] (12 8 10)

Figures:

Square [5 7] (3)
Rectangle [2 4] [8 6]
Trapeze [3 6] (7 4 5.5)

5. Результаты выполнения тестов:

user@PSB133S01ZFH:~/3sem_projects/oop_exercise_03/tests\$ bash test.sh ../build/lab3

Test test_00.test: SUCCESS

Test test_01.test: SUCCESS

6. Объяснение результатов работы программы:

- 1) Программа создаёт вектор — там мы будем хранить данные о фигурах;
- 2) Каждая структура геометрических фигур (square, rectangle, trapeze) является потомком структуры figure (центр тяжести, площадь, вывод данных о фигуре), которая является потомком структуры point (координаты x и y);
- 3) Программа выполняет команды в виде чисел:
 - A) 0 — выход;
 - B) 1 — создание элемента и добавление в него данных об определённой фигуре:
 - I. 0 — квадрат (вершина, сторона);
 - II. 1 — прямоугольник (вершина 1, вершина 2);
 - III. 2 — трапеция (вершина, большее основание, меньшее основание, высота)
 - C) 2 — удаление элемента по его индексу;

- D) 3 — вывести координаты центров тяжести всех элементов
- E) 4 — вывести площади всех элементов
- F) 5 — вывести введенные данные всех элементов
- 4) При выходе программа удаляет вектор со всеми элементами.

7. **Вывод:** 1) Ознакомились с наследованием и полиморфизмом в C++ и усвоили навык работы с ними; 2) Написана программа, производящая операции с помощью наследования и полиморфизма.