Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Отчёт по заданию № 1**

**на тему: “Работа с иерархией объектов: наследование и полиморфизм”**

**по дисциплине**

**“Алгоритмы и структуры данных”**

**Вариант 26**

|  |  |
| --- | --- |
| Выполнил студент гр.9308: | Яловега Н.В. |
| Проверил: | Колинько П.Г. |

Санкт-Петербург, 2020 г.

**Оглавление**

[​ Введение 3](#__RefHeading___Toc304_2067246640)

[​ 1. Задание 4](#__RefHeading___Toc306_2067246640)

[​ 2. Формализация задания 5](#__RefHeading___Toc308_2067246640)

[3. Работа с наследованием объектов 6](#__RefHeading___Toc318_2067246640)

[​ 4. Результаты эксперимента 7](#__RefHeading___Toc320_2067246640)

[​ Вывод 10](#__RefHeading___Toc326_2067246640)

[​ Список используемых источников 11](#__RefHeading___Toc328_2067246640)

[​ Приложение 1 (Исходный текст программы) 12](#__RefHeading___Toc330_2067246640)

## Введение

Целью задания является ознакомление с принципами ООП, в частности, с принципами наследования и полиморфизма, использование библиотеки «из пакета» и последующего расширения его функционала.

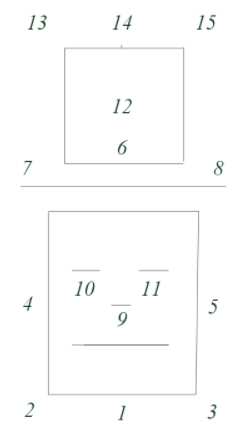
## 1. Задание

Доработать модуль shape.cpp, добавив в коллекцию еще одну фигуру — трапецию с крестом. Для этой фигуры нужно определить подходящее место в иерархии классови написать недостающие функции-члены.

Разработанной фигурой нужно дополнить картинку в позициях 7, 8 - рога, 14 - шишак. Необходимо написать функции, обеспечивающие примыкание очередной фигуры p с нужной стороны по отношению к уже размещенной q.

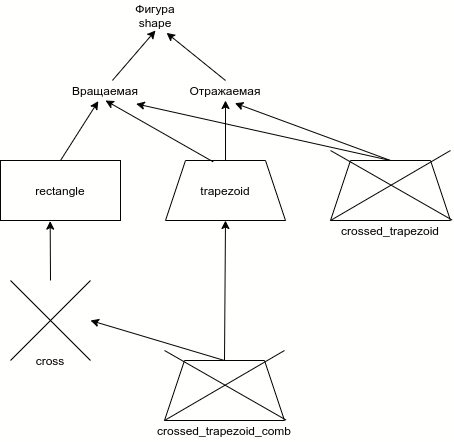
## 2. Формализация задания

* + 1. Имеется схема объекта с пометками возможных точек присоединения (рис.1), в нашему случае такими точками будут: 7, 8, 14.



*Рис.1: Возможные точки стыковки*

Фигурой для присоединения будет использоваться трапеция с крестом. Фигуру можно реализовать несколькими способами. Т.к фигура — состовная, то можем реализовать отдельно классы для трапеции и косого креста. Тогда класс трапеции с крестом будет наследоваться от этих классов. Либо можем рассматривать фигуру как единое целое и реализовать ее при помощи наследования от классов rotatable и reflectable. Эти идеи отображены на рисунке 2.

Рис 2: Иерархия классов фигур

## 3. Работа с наследованием объектов

**3.1. Добавленные классы**

Для реализации фигуры трапеции с косым крестом был создан класс crossed\_trapezoid, являющийся производным от классов rotatable и reflectable. Наследование от данных классов необходимо т.к. фигура в общем случае не является симметричной.

Также для реализации комбинорованной фигуры трапеции с косым крестом был создан класс crossed\_trapezoid\_comb, являющийся производным от классов trapezoid и cross. Класс trapezoid реализует фигуру трапецию и является производным от классов rotatable и reflectable, т.к. фигура в общем случае не является симметричной. Класс cross реализует фигуру косого креста и является производным от класса rectangle.

**3.2. Переопределение функций-членов у добавленного класса**

Для класса cross пришлось переопределить метод рисования фигуры draw. Все остальные методы наследуются от rectangle.

Для класса trapezoid пришлось переопределить методы вычисления точек привязки, рисования, изменения размера, перемещения, поворот фигуры вправо и влево, отражение фигуры.

Для класса crossed\_trapezoid пришлось переопределить методы вычисления точек привязки, рисования, изменения размера, перемещения, поворот фигуры вправо и влево, отражение фигуры.

Для класса crossed\_trapezoid\_comb пришлось переопределить методы вычисления точек привязки, рисования, изменения размера, перемещения, поворот фигуры вправо и влево.

**3.3. Реализация недоступности необязательных функций-членов класса**

Было решено сделать недоступными четыре функции-члена: конструктор копирования, конструктор перемещения, оператор присваивания копированием и оператор присваивания перемещением.

Способом реализации было выбрано изменение модификаторов доступа функций-членов на приватный. Таким образом, при попытке воспользоваться этими функциями возникнет ошибка на этапе компиляции программы.

## 4. Результаты эксперимента

Для начала убедимся в корректности создания и отрисовки фигур :



*Рис.3. Отрисовка исходных фигур*

Как видно из рисунка 3, фигуры отрисовываются корректно. При полученнии результата нужно учитывать неидеальность принятой системы отображений фигур на экране и использование целых чисел для координат точек.

Сформируем левый и правый рога из трапеций с крестом путем их увеличения в 2 разa и поворота их вправо и влево соответственно. Шишак сформируем из комбинированной трапеции с крестом путем отражения фигуры горизотально:

*Рис.4. Преобразование фигур*

Как видно из рисунка 4, фигуры увеличиваются ,отражаются и поворачиваются корректно.

Уже подготовленные фигуры состыкуем на соответствующие позиции, используя функции, реализованые состыковки фигур в разных позициях. Поле шляпы должно размещаться над лицом, шляпа должна находится над полем шляпы, шишак должен располагаться над шляпой, правый рог должен быть расположен справа над полем шляпы, левый рог должен быть расположен слева над полем шляпы :

*Рис.5. Преобразование фигур*

Как видно из рисунка 5, фигуры размещаются корректно.

## Вывод

При выполнении данной работы были получены практические навыки по работе с наследованием классов. Были изучены понятия производного класса, полиморфизма, виртуальных классов, механизмы работы с классами.

## Список используемых источников

1. Колинько П.Г. Пользовательские контейнеры / Методические указания по дисциплине «Алгоритмы и структуры данных» - Санкт-Петербург: СПбГЭТУ «ЛЭТИ», 2020.
2. Урок №163. Виртуальные функции и Полиморфизм. URL: <https://ravesli.com/urok-163-virtualnye-funktsii-i-polimorfizm/>
3. Урок №114. Спецификаторы доступа public и private. [https://ravesli.com/urok-114-spetsifikatory-dostupa-public-i-private](https://ravesli.com/urok-114-spetsifikatory-dostupa-public-i-private/)/

## Приложение 1 (Исходный текст программы)

/\*

Пополнение и использование библиотеки фигур

Добавленные фигуры:

- trapezoid: трапеция

- cross: косой крест

- crossed\_trapezoid\_comb: трапеция с косым крестом, реализованная наследованием

предыдущих фигур

- crossed\_trapezoid: трапеция с косым крестом

Добавленные функции размещения:

- down(p, q): поместить p над q

- left\_up(p, q): поместить p слева над q

- right\_up(p, q): поместить p справа над q

- right\_down(p, q): поместить p справа под q

- left\_down(p, q): поместить p справа под q

\*/

#include "screen.h"

#include "shape.h"

using namespace std;

// ========== ДОБАВКА ============

/\*

Реализация фигуры трапеция

\*/

class trapezoid : public rotatable, public reflectable

/\*

nw b------n------c ne

/ \

/ \

w / \ e

/ \

/ \

a-----------s------------d

sw se

\*/

{

trapezoid(const trapezoid&);

trapezoid(const trapezoid&&);

trapezoid& operator=(const trapezoid &) = delete;

trapezoid& operator=(trapezoid &&) = delete;

protected:

point a, b, c, d;

public:

trapezoid(point, int, point, int);

point north() const { return point((swest().x + seast().x) / 2, nwest().y); }

point south() const { return point((swest().x + seast().x) / 2, swest().y); }

point east() const { return point(seast().x, (neast().y + seast().y) / 2); }

point west() const { return point(swest().x, (swest().y + nwest().y) / 2); }

point neast() const { return point(max(a.x, max(b.x, max(c.x, d.x))), max(a.y, max(b.y, max(c.y, d.y)))); }

point seast() const { return point(max(a.x, max(b.x, max(c.x, d.x))), min(a.y, min(b.y, min(c.y, d.y)))); }

point nwest() const { return point(min(a.x, min(b.x, min(c.x, d.x))), max(a.y, max(b.y, max(c.y, d.y))));}

point swest() const { return point(min(a.x, min(b.x, min(c.x, d.x))), min(a.y, min(b.y, min(c.y, d.y)))); }

void rotate\_left();

void rotate\_right();

void flip\_horisontally();

void flip\_vertically();

void move(int, int);

void resize(int);

void draw();

};

trapezoid :: trapezoid (point a\_, int lena, point b\_, int lenb)

/\*

Конструктор принимает точку sw и длину основания,

точку nw и длину основания. Этот набор данных может

задавать трапецию любого вида.

Для хранения вычисляются координаты 4 точек.

\*/

{

a = a\_;

b = b\_;

c.x = b.x + lenb; c.y = b.y;

d.x = a.x + lena; d.y = a.y;

}

void trapezoid::rotate\_right()

{

int x0 = (neast().x + seast().x + nwest().x + swest().x)/4;

int y0 = (neast().y + seast().y + nwest().y + swest().y)/4;

int x, y;

x = a.x; y = a.y;

a.x = x0 + (y - y0)\*2;

a.y = y0 - (x - x0)/2;

x = b.x; y = b.y;

b.x = x0 + (y - y0)\*2;

b.y = y0 - (x - x0)/2;

x = c.x; y = c.y;

c.x = x0 + (y - y0)\*2;

c.y = y0 - (x - x0)/2;

x = d.x; y = d.y;

d.x = x0 + (y - y0)\*2;

d.y = y0 - (x - x0)/2;

}

void trapezoid::rotate\_left()

{

int x0 = (neast().x + seast().x + nwest().x + swest().x)/4;

int y0 = (neast().y + seast().y + nwest().y + swest().y)/4;

int x, y;

x = a.x; y = a.y;

a.x = x0 - (y - y0)\*2;

a.y = y0 + (x - x0)/2;

x = b.x; y = b.y;

b.x = x0 - (y - y0)\*2;

b.y = y0 + (x - x0)/2;

x = c.x; y = c.y;

c.x = x0 - (y - y0)\*2;

c.y = y0 + (x - x0)/2;

x = d.x; y = d.y;

d.x = x0 - (y - y0)\*2;

d.y = y0 + (x - x0)/2;

}

void trapezoid :: flip\_horisontally()

{

int dx1 = abs(b.x - nwest().x),

dx2 = abs(c.x - nwest().x),

dx3 = abs(a.x - swest().x),

dx4 = abs(d.x - swest().x);

a.x = swest().x + dx1;

b.x = nwest().x + dx3;

c.x = nwest().x + dx4;

d.x = swest().x + dx2;

}

void trapezoid :: flip\_vertically()

{

rotate\_right();

rotate\_right();

flip\_horisontally();

}

void trapezoid :: move(int dx, int dy)

{

a.x += dx; a.y += dy;

b.x += dx; b.y += dy;

c.x += dx; c.y += dy;

d.x += dx; d.y += dy;

}

void trapezoid :: resize(int scale)

{

b.y += (b.y - a.y) \* (scale - 1);

d.x += (d.x - a.x) \* (scale - 1);

c.x += (c.x - b.x) \* (scale - 1);

c.y += (c.y - d.y) \* (scale - 1);

}

void trapezoid :: draw()

{

put\_line(a, b);

put\_line(b, c);

put\_line(c, d);

put\_line(a, d);

}

/\*

Реализация фигуры косой крест

\*/

class cross : public rectangle

/\*

nw n ne

\ /

\ /

w c e

/ \

/ \

sw s se

\*/

{

cross(const cross&);

cross(const cross&&);

cross& operator=(const cross &) = delete;

cross& operator=(cross &&) = delete;

public:

cross(point a, point b) : rectangle (a,b) {}

void draw();

};

void cross :: draw()

{

put\_line(rectangle::swest(), rectangle::neast());

put\_line(rectangle::nwest(), rectangle::seast());

}

/\*

Реализация трапеции с косым крестом наследованием двух фигур (трапеции и креста)

\*/

class crossed\_trapezoid\_comb : public trapezoid, public cross

{

crossed\_trapezoid\_comb(const crossed\_trapezoid\_comb&);

crossed\_trapezoid\_comb(const crossed\_trapezoid\_comb&&);

crossed\_trapezoid\_comb& operator=(const crossed\_trapezoid\_comb &) = delete;

crossed\_trapezoid\_comb& operator=(crossed\_trapezoid\_comb &&) = delete;

public:

crossed\_trapezoid\_comb(point a, int lena, point b, int lenb):

trapezoid(a, lena, b, lenb), cross(a, point(a.x + lena, b.y)) {}

point north() const{ return cross::north(); } // север

point south() const{ return cross::south(); } // юг

point east() const{ return cross::east(); } // восток

point west() const{ return cross::west(); } // запад

point neast() const{ return cross::neast(); } // северо-восток

point seast() const{ return cross::seast(); } // юго-восток

point nwest() const{ return cross::nwest(); } // северо-запад

point swest() const{ return cross::swest(); } // северо-восток

void rotate\_left();

void rotate\_right();

void move(int, int);

void resize(int);

void draw();

};

void crossed\_trapezoid\_comb::rotate\_left()

{

trapezoid::rotate\_left();

cross::rotate\_left();

}

void crossed\_trapezoid\_comb::rotate\_right()

{

trapezoid::rotate\_right();

cross::rotate\_right();

}

void crossed\_trapezoid\_comb::move(int dx, int dy)

{

trapezoid::move(dx, dy);

cross::move(dx, dy);

}

void crossed\_trapezoid\_comb::resize(int scale)

{

trapezoid::resize(scale);

cross::resize(scale);

}

void crossed\_trapezoid\_comb::draw()

{

cross::draw();

trapezoid::draw();

}

/\*

Реализация трапеции с косым крестом без наследования от отдельных фигур

\*/

class crossed\_trapezoid : public rotatable, public reflectable

{

crossed\_trapezoid(const crossed\_trapezoid&);

crossed\_trapezoid(const crossed\_trapezoid&&);

crossed\_trapezoid& operator=(const crossed\_trapezoid &) = delete;

crossed\_trapezoid& operator=(crossed\_trapezoid &&) = delete;

protected:

point a, b, c, d;

public:

crossed\_trapezoid(point, int, point, int);

point north() const { return point((swest().x + seast().x) / 2, nwest().y); }

point south() const { return point((swest().x + seast().x) / 2, swest().y); }

point east() const { return point(seast().x, (neast().y + seast().y) / 2); }

point west() const { return point(swest().x, (swest().y + nwest().y) / 2); }

point neast() const { return point(max(a.x, max(b.x, max(c.x, d.x))), max(a.y, max(b.y, max(c.y, d.y)))); }

point seast() const { return point(max(a.x, max(b.x, max(c.x, d.x))), min(a.y, min(b.y, min(c.y, d.y)))); }

point nwest() const { return point(min(a.x, min(b.x, min(c.x, d.x))), max(a.y, max(b.y, max(c.y, d.y))));}

point swest() const { return point(min(a.x, min(b.x, min(c.x, d.x))), min(a.y, min(b.y, min(c.y, d.y)))); }

void rotate\_left();

void rotate\_right();

void flip\_horisontally();

void flip\_vertically();

void move(int, int);

void resize(int);

void draw();

};

crossed\_trapezoid :: crossed\_trapezoid (point a\_, int lena, point b\_, int lenb)

/\*

Конструктор принимает точку sw и длину основания,

точку nw и длину основания. Этот набор данных может

задавать трапецию любого вида.

Для хранения вычисляются координаты 4 точек.

\*/

{

a = a\_;

b = b\_;

c.x = b.x + lenb; c.y = b.y;

d.x = a.x + lena; d.y = a.y;

}

void crossed\_trapezoid::rotate\_right()

{

int x0 = (neast().x + seast().x + nwest().x + swest().x)/4;

int y0 = (neast().y + seast().y + nwest().y + swest().y)/4;

int x, y;

x = a.x; y = a.y;

a.x = x0 + (y - y0)\*2;

a.y = y0 - (x - x0)/2;

x = b.x; y = b.y;

b.x = x0 + (y - y0)\*2;

b.y = y0 - (x - x0)/2;

x = c.x; y = c.y;

c.x = x0 + (y - y0)\*2;

c.y = y0 - (x - x0)/2;

x = d.x; y = d.y;

d.x = x0 + (y - y0)\*2;

d.y = y0 - (x - x0)/2;

}

void crossed\_trapezoid::rotate\_left()

{

int x0 = (neast().x + seast().x + nwest().x + swest().x)/4;

int y0 = (neast().y + seast().y + nwest().y + swest().y)/4;

int x, y;

x = a.x; y = a.y;

a.x = x0 - (y - y0)\*2;

a.y = y0 + (x - x0)/2;

x = b.x; y = b.y;

b.x = x0 - (y - y0)\*2;

b.y = y0 + (x - x0)/2;

x = c.x; y = c.y;

c.x = x0 - (y - y0)\*2;

c.y = y0 + (x - x0)/2;

x = d.x; y = d.y;

d.x = x0 - (y - y0)\*2;

d.y = y0 + (x - x0)/2;

}

void crossed\_trapezoid :: flip\_horisontally()

{

int dx1 = abs(b.x - nwest().x),

dx2 = abs(c.x - nwest().x),

dx3 = abs(a.x - swest().x),

dx4 = abs(d.x - swest().x);

a.x = swest().x + dx1;

b.x = nwest().x + dx3;

c.x = nwest().x + dx4;

d.x = swest().x + dx2;

}

void crossed\_trapezoid :: flip\_vertically()

{

rotate\_right();

rotate\_right();

flip\_horisontally();

}

void crossed\_trapezoid :: move(int dx, int dy)

{

a.x += dx; a.y += dy;

b.x += dx; b.y += dy;

c.x += dx; c.y += dy;

d.x += dx; d.y += dy;

}

void crossed\_trapezoid :: resize(int r)

{

b.y += (b.y - a.y) \* (r - 1);

d.x += (d.x - a.x) \* (r - 1);

c.x += (c.x - b.x) \* (r - 1);

c.y += (c.y - d.y) \* (r - 1);

}

void crossed\_trapezoid :: draw()

{

put\_line(a, b);

put\_line(b, c);

put\_line(c, d);

put\_line(a, d);

put\_line(swest(), neast());

put\_line(nwest(), seast());

}

/\*

Функции размещения фигур относительно друг друга

\*/

void down(shape& p, const shape& q)

// Поместить p над q

{

p.move(q.south().x - p.north().x, q.south().y - p.north().y - 1);

}

void left\_up(shape& p, const shape& q)

// Поместить p слева над q

{

p.move(q.nwest().x - p.swest().x, q.nwest().y - p.swest().y + 1);

}

void right\_up(shape& p, const shape& q)

// Поместить p справа над q

{

p.move(q.neast().x - p.seast().x, q.nwest().y - p.swest().y + 1);

}

void right\_down(shape& p, const shape& q)

// Поместить p справа под q

{

p.move(q.east().x - p.west().x, q.swest().y - p.nwest().y);

}

void left\_down(shape& p, const shape& q)

// Поместить p справа под q

{

p.move(q.west().x - p.east().x, q.swest().y - p.nwest().y);

}

// ========== КОНЕЦ ДОБАВКИ ============

class face: public rectangle

{

int w, h;

line l\_eye, r\_eye, mouth;

public:

face(point, point);

void draw();

void move(int, int);

void resize(int) {};

};

face :: face (point a, point b):

rectangle(a, b),

w(neast().x - swest().x + 1),

h(neast().y - swest().y + 1),

l\_eye(point(swest().x + 2, swest().y + h \* 3 / 4), 2),

r\_eye(point(swest().x + w - 4, swest().y + h \* 3 / 4), 2),

mouth(point(swest().x + 2, swest().y + h / 4), w - 4)

{}

void face :: draw()

{

rectangle :: draw();

int a = (swest().x + neast().x)/2;

int b = (swest().y + neast().y)/2;

put\_point(point(a, b));

}

void face :: move(int a, int b)

{

rectangle :: move(a, b);

l\_eye.move(a, b);

r\_eye.move(a, b);

mouth.move(a, b);

}

int main()

{

screen\_init();

// объявление набора фигур

face f(point(49, 1), point(71, 16));

line brim(point(5, 18), 17);

rectangle hat(point(55, 20), point(69, 25));

crossed\_trapezoid right\_horn(point(10, 25), 10, point(10, 28), 6);

crossed\_trapezoid left\_horn(point(10, 5), 10, point(10, 8), 6);

crossed\_trapezoid\_comb shishak(point(80, 5), 16, point(85, 10), 6);

shape\_refresh();

std::cout << "=== Generated... ===\n";

std::cin.get(); //Смотреть исходный набор

// подготовка к сборке

hat.rotate\_right();

hat.resize(2);

brim.resize(4);

right\_horn.resize(2);

right\_horn.rotate\_left();

left\_horn.resize(2);

left\_horn.flip\_vertically();

left\_horn.rotate\_right();

shishak.flip\_horisontally();

shape\_refresh();

std::cout << "=== Prepared... ===\n";

std::cin.get(); //Смотреть результат поворотов/отражений

// сборка изображения

up(brim, f);

up(hat, brim);

right\_up(right\_horn, brim);

left\_up(left\_horn, brim);

up(shishak, hat);

shape\_refresh();

std::cout << "=== Ready! ===\n";

std::cin.get(); //Смотреть результат

screen\_destroy();

return 0;

}