**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 1**

**по дисциплине «АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ»**

**Тема: Работа с иерархией объектов:  
Наследование и полиморфизм**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3312 |  | Нукусси Фруктье |
|  |  |  |
| Преподаватель |  | Колинько П.Г. |

Санкт-Петербург

2025

Описание получившейся иерархии классов и пояснения:

1) Добавленные классы:

-класс **cross**, который будет создавать наш крест

2) Пришлось переопределить три функции:

**draw**, добавили рисование двух линий

**move**, добавили перемещений двух линий

**resize**, добавили изменение размера двух линий

3)Не было необходимости делать недоступными функции-члены

В файле shape.h:

#include <list>

class cross : public shape {

protected:

point center;

int size;

public:

*// Constructeur*

cross(point c, int s)

: center(c), size(s) { }

*// Redéfinition des méthodes virtuelles pures*

point north() const override { return point(center.x, center.y - size / 2); }

point south() const override { return point(center.x, center.y + size / 2); }

point east() const override { return point(center.x + size / 2, center.y); }

point west() const override { return point(center.x - size / 2, center.y); }

point neast() const override { return point(center.x + size / 2, center.y - size / 2); }

point seast() const override { return point(center.x + size / 2, center.y + size / 2); }

point nwest() const override { return point(center.x - size / 2, center.y - size / 2); }

point swest() const override { return point(center.x - size / 2, center.y + size / 2); }

void draw() override {

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

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

}

void move(int dx, int dy) override {

center.x += dx;

center.y += dy;

}

void resize(double factor) override {

size = static\_cast<int>(size \* factor);

}

};

void push\_vert(shape& p,int d )

{

p.move(0, d);

}

void push\_side(shape& p,int d )

{

p.move(d, 0);

}

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

{ point n = q.seast( );

point s = p.west( );

p.move(n.x - s.x + 2, n.y - s.y ); }

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

{ point n = q.swest( );

point s = p.east( );

p.move(n.x - s.x -2, n.y - s.y ); }

**файл shape.cpp:**

int main( )

{ setlocale(LC\_ALL, "Rus");

screen\_init( );

*//== 1. Объявление набора фигур ==*

cross myCross1(point(8,3),3);

cross myCross2(point(34,3),3);

cross myCross3(point(24,9),3);

cross myCross4(point(12,6),3);

rectangle hat(point(0, 0), point(14, 5));

*//cross my\_cross(point(21,5),3);*

line brim(point(20,9),17);

myshape face(point(15,10), point(27,18));

h\_circle beard(point(40,10), 5);

shape\_refresh( );

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

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

*//== 2. Подготовка к сборке ==*

hat.rotate\_right( );

brim.resize(2.0);

*//face.resize(1.2);*

face.resize(1.5);

beard.flip\_vertically( );

beard.resize(1.5);

shape\_refresh( );

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

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

*//== 3. Сборка изображения ==*

face.move(0, -8); *// Лицо – в исходное положение (если нужно!)*

up(brim, face);

up(hat, brim);

down(beard, face);

left(myCross1, beard);

right(myCross2, beard);

push\_vert(myCross2, 10);

push\_vert(myCross1, 10);

push\_side(myCross1, -2);

push\_side(myCross2, 2);

push\_side(myCross3, -6);

push\_vert(myCross3, -3);

push\_side(myCross4, 13);

(myCross1, beard);

shape\_refresh( );

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

std::cin.get(); *//Смотреть результат*

screen\_destroy( );

return 0;

}

