**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.

|  |  |  |
| --- | --- | --- |
| Студенты гр. 0308 |  | Бобыльков Т.В. |
|  |  | Радабольский В. C. |
| Преподаватель |  | Манирагена В. |

Санкт-Петербург

2022

**Цель работы.**

Целью работы является исследование принципов объектно-ориентированного подхода в программировании, а именно полиморфизма и наследование, а также создание программы для рисования фигур согласно варианту.

**Задание (вариант 33)**

Дополнить рисунок фигурой «крест» (10) фигуру «физиономия» в местах «уши» (4, 5) и «эмблема на шляпе» (12).

**Решение задания.**

Для дополнения рисунка, согласно заданию, добавим в программу класс фигуры «крест», используя механизм наследования. Класс «cross» будет наследовать класс абстрактной фигуры «shape» в открытом виде (public), потому что фигура «крест» полностью является фигурой.

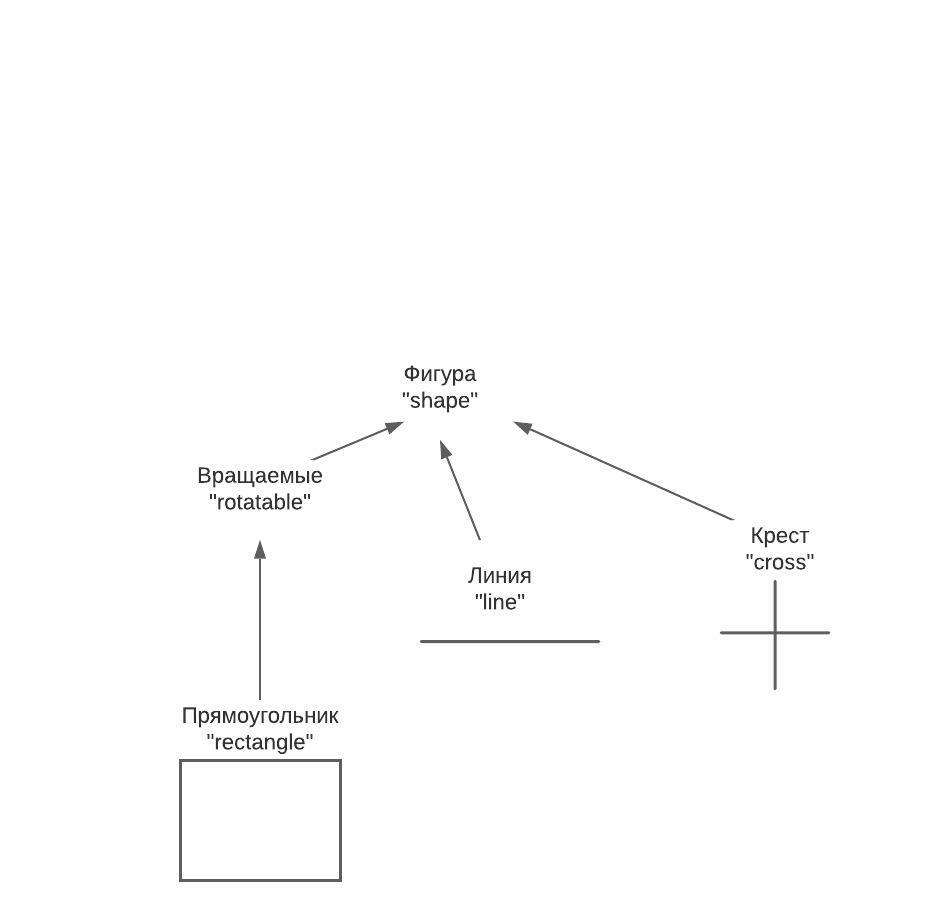
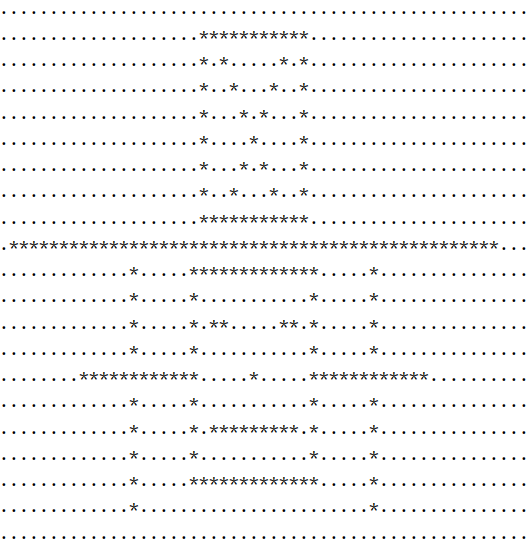
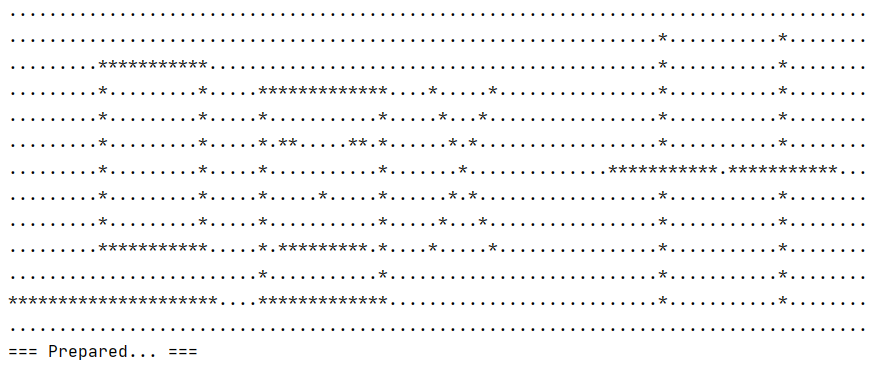
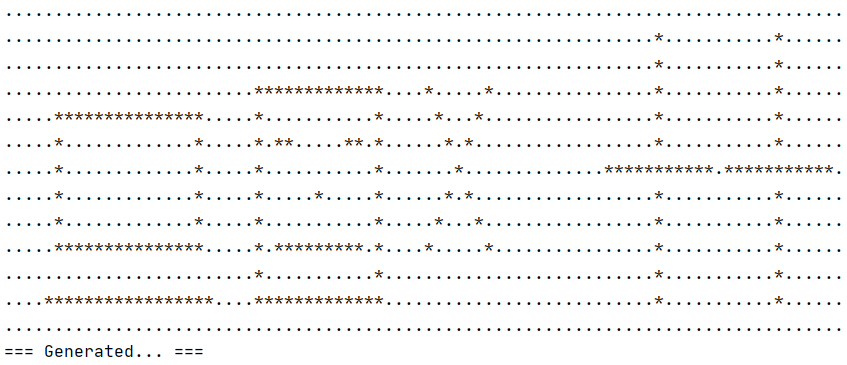
Из программы, представленной в методических указаниях не было удалено и\или переопределено никаких классов, потому что они подходили структуре нашей программы.

Рисунок . Иерархия объектов

Для нового класса крест были переопределены такие функции как draw (рисование), resize (изменение размера) и move (перемещение), а также все функции, возвращающие позиционные точки, потому что класс было решено воспринимать как центровую точку (point c\_center) с фиксированным размером «лопастей» (int size).  
**Контрольный тест** 

**Выводы.**

В ходе выполнения работы мы научились пользоваться такими принципами ООП как полиморфизм и наследование, что помогло успешно модернизировать код, указанный в методических указаниях. Были изучены динамические связывания для переопределения одноименных функций классов-наследников. Мы на собственном опыте смоделировали ситуацию, в которой объектно-ориентированный подход имеет высокое преимущество – дополнение уже существующих программ. Прямо на защите нам удалось за 10 строк кода создать новый класс андреевского креста, который наследует класс обычного креста.

**Список использованных источников.**

П. Г. Колинько, Алгоритмы и структуры данных. Лекция от 07.02.2022.

Методическое пособие «Пользовательские Контейнеры» П.Г. Колинько.

**Приложение.**

Исходный код программы для решения задачи на языке C++:

#include <iostream>  
#include "screen.h"  
#include "shape.h"  
  
  
class cross : public shape {  
 point c\_center;  
 int size;  
public:  
 cross(point c, int s): c\_center(c), size(s) {};  
 point north() const {return point(c\_center.x, c\_center.y + size);}; *//точки для привязки* point south() const {return point(c\_center.x, c\_center.y - size);};  
 point west() const {return point(c\_center.x - size, c\_center.y);};  
 point east() const {return point(c\_center.x + size, c\_center.y);};  
  
 point neast() const {return point(c\_center.x + size, c\_center.y + size);};  
 point seast() const {return point(c\_center.x + size, c\_center.y - size);};  
 point nwest() const {return point(c\_center.x - size, c\_center.y + size);};  
 point swest() const {return point(c\_center.x - size, c\_center.y - size);};  
  
 point center() const {return c\_center;};  
  
  
 void move(int a, int b); *// перемещение* void resize(double d); *// Изменение размера* void draw(); *// переопределим рисование*};

class andrew\_cross : public cross {  
  
public:  
 andrew\_cross(point c, int s) {  
 c\_center = c;  
 size = s;  
 };  
 void draw();  
};  
void andrew\_cross::draw() {  
 put\_line(neast(), swest());  
 put\_line(nwest(), seast());  
};

void cross ::draw() {  
 put\_line(north(), south());  
 put\_line(east(), west());  
}  
  
void cross::resize(double d) {  
 size \*= d;  
}  
  
void cross::move(int a, int b) {  
 c\_center.x += a;  
 c\_center.y += b;  
}  
  
  
*// в центр*void into(shape &p, const shape &q) {  
 point p\_c = p.center();  
 point q\_c = q.center();  
  
 p.move(q\_c.x - p\_c.x, q\_c.y - p\_c.y);  
}  
*// слева*void left(shape &p, const shape &q) {  
 point e = q.west();  
 point w = p.east();  
 p.move(e.x - w.x - 1, e.y - w.y);  
}  
*// справа*void right(shape &p, const shape &q) {  
 point w = q.east();  
 point e = p.west();  
 p.move(w.x - e.x + 1, w.y - e.y);  
}  
  
class myshape : public rectangle {  
 int w, h;  
 line l\_eye; *// Левый глаз* line r\_eye; *// Правый глаз* line mouth; *// рот*public:  
 myshape(point, point);  
 void draw() ;  
 void move(int, int);  
 void resize(double) {}  
};  
  
myshape ::myshape(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 myshape :: draw() {  
 rectangle :: draw(); *// Контур лица* point nose = center();  
  
 put\_point(nose);  
}  
  
void myshape :: 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() {  
  
 setlocale(**LC\_ALL**, "Rus");  
  
 screen\_init();  
  
 *// ==1. Объявление набор фигур ==* rectangle hat(point(5,3), point(5+14,3+5));  
 line brim(point(4,1), 17); *// козырек* myshape face(point(25,1), point(25+12,1+8));  
 cross left\_cross(point(65, 6), 5);  
 cross right\_cross(point(77, 6), 5);  
 andrew\_cross hat\_cross(point(45, 6), 3);  
  
 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(2.0);  
  
 shape\_refresh( );  
 std::cout << "=== Prepared... ===\n";  
 std::cin.get(); *//Смотреть результат поворотов/отражений  
//== 3.Сборка изображения ==* face.move(10, 10); *// Лицо - в исходное положение* up(brim, face);  
 up(hat, brim);  
 into(hat\_cross, hat);  
 left(left\_cross, face);  
 right(right\_cross, face);  
 shape\_refresh( );  
 std::cout << "=== Ready! ===\n";  
  
 return 0;  
}