**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ.УЛЬЯНОВА (ЛЕНИНА)**

Кафедра вычислительной техники

Отчет по лабораторной работе №1 на тему:

«Работа с иерархией объектов:  
наследование и полиморфизм»

по дисциплине «Алгоритмы и структуры данных»

Колинько П.Г.

Санкт-Петербург

2018

Торопов В.А.

Преподаватель

Студент гр. 7307

Оглавление

[Цель работы 3](#_Toc2171479)

[Задание 3](#_Toc2171480)

[Иерархия Классов 4](#_Toc2171481)

[Код программы 5](#_Toc2171482)

# **Цель работы**

Получить навыки работы с иерархией классов.

# **Задание**

Доработать модуль shape.cpp: добавить в коллекцию фигуру полукруг. Для этой фигуры нужно будет оп­ределить подходящее место в иерархии классов и написать необходимые функ­ции-члены. Функции-члены, использование которых не предполагается, можно объявить так, чтобы они были недоступны. Поместить фигуру в позиции, указанных цифрами 7, 8, 2, 3 (рис. 1) в виде «рогов» и «свисающих усов» с помощью функций отражения.



*Рис*. *1.* Позиции на результате работы программы  
 для возможной вставки дополнительной фигуры

# Иерархия Классов

Фигура (shape)

Вращаемая (rotatable)

Отражаемая (reflectable)

Линия (line)

Прямоугольник(rectangle)

Полукруг

(h\_circle)

Моя фигура

(my\_shape)

Функции-члены rotate\_right(), rotate\_left() сделаны не доступными путём переопределения функций в классе h\_circle;

# **Код программы**

#include "pch.h"

#include <iostream>

#include <locale.h>

#include "screen.h"

#include "shape.h"

// Пример добавки: дополнительный фрагмент - полуокружность

class h\_circle : public rectangle, public reflectable{

bool reflected;

public:

h\_circle(point a, point b, bool r = true) : rectangle(a, b), reflected(r) { }

void draw();

void flip\_horisontally() { }; // Отразить горизонтально

void flip\_vertically() { reflected = !reflected; }; // Отразить вертикально

void rotate\_right() { }; //my

void rotate\_left() { }; //my

};

void h\_circle::draw() //Алгоритм Брезенхэма для окружностей

{

int x0 = (sw.x + ne.x) / 2;

int y0 = reflected ? sw.y : ne.y;

int radius = (ne.x - sw.x) / 2;

int x = 0;

int y = radius;

int delta = 2 - 2 \* radius;

int error = 0;

while (y >= 0) {

if (reflected) { put\_point(x0 + x, y0 + y \* 0.7); put\_point(x0 - x, y0 + y \* 0.7); }

else { put\_point(x0 + x, y0 - y \* 0.7); put\_point(x0 - x, y0 - y \* 0.7); }

error = 2 \* (delta + y) - 1;

if (delta < 0 && error <= 0) { ++x; delta += 2 \* x + 1; continue; }

error = 2 \* (delta - x) - 1;

if (delta > 0 && error > 0) { --y; delta += 1 - 2 \* y; continue; }

++x; delta += 2 \* (x - y); --y;

}

}

// Пример добавки: дополнительная функция присоединения…

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

{

point n = q.south();

point s = p.north();

p.move(n.x - s.x, n.y - s.y - 1);

}

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

{

point w = p.swest();

point ne = q.neast();

p.move(ne.x - w.x + 1, ne.y - w.y);

}

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

{

point e = p.seast();

point nw = q.nwest();

p.move(nw.x - e.x, nw.y - e.y);

}

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

{

point nw = p.nwest();

point se = q.seast();

p.move(se.x - nw.x + 1, se.y - nw.y);

}

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

{

point ne = p.neast();

point sw = q.swest();

p.move(sw.x - ne.x, sw.y - ne.y);

}

// Cборная фигура - физиономия

class myshape : public rectangle {

//Моя фигура ЯВЛЯЕТСЯ прямоугольником

int w, h;

line l\_eye; // левый глаз – моя фигура СОДЕРЖИТ линию

line r\_eye; // правый глаз

line mouth; // рот

public:

myshape(point, point);

void draw();

void move(int, int);

void resize(int) { }

};

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();

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

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

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

}

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() //только для Visual C++ (иначе – main( ))

{

setlocale(LC\_ALL, "Rus");

screen\_init();

//== 1.Объявление набора фигур ==

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

line brim(point(0, 15), 8); //17

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

h\_circle beard(point(40, 10), point(50, 20));

h\_circle rog1(point(70, 10), point(80, 20)); //my

h\_circle rog2(point(50, 10), point(60, 20)); //my

h\_circle rog3(point(10, 10), point(15, 15)); //my

h\_circle rog4(point(30, 10), point(35, 15)); //my

shape\_refresh();

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

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

//== 2.Ориентация ==

hat.rotate\_right();

brim.resize(2);

face.resize(2);

beard.flip\_vertically();

rog3.flip\_vertically();//my

rog4.flip\_vertically();//my

shape\_refresh();

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

std::cin.get(); //Смотреть ориентацию

//== 3.Сборка изображения ==

// face.move(0, -10); // В исходное положение

up(brim, face);

up(hat, brim);

down(beard, face);

right\_up(rog1, face);

left\_up(rog2, face);

right\_down(rog3, face);

left\_down(rog4, face);

shape\_refresh();

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

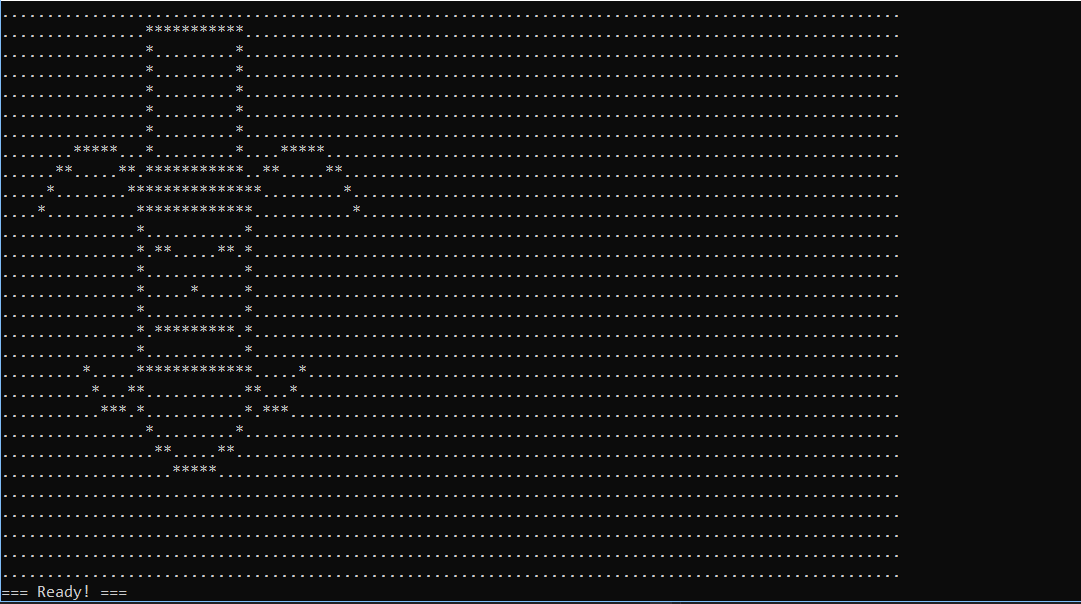
std::cin.get(); //Смотреть результат

screen\_destroy();

return 0;

}

# Пример работы программы



# Вывод

В ходе работы были получены навыки в работе с иерархией классов. Благодаря наследованию удалось сократить количество кода при написании новых фигур. Было применено виртуальное наследование, чтобы не создавать лишних экземпляров родительского класса, которые в дальнейшем могли вызвать проблемы. Благодаря полиморфизму функции способны обрабатывать каждую фигуру единообразно, что, опять же сокращает количество кода. Удачно создан класс h\_circle. Благодаря одной булевой переменной можно отражать полукруг.