

МИНОБРАЗОВАНИЯ 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, часть 2»
Тема: «Работа с иерархией объектов: наследование и
полиморфизм»

Студенты гр. 9306

Евдокимов О.В. Кныш С.А. Павельев М.С.

Преподаватель

Манерагена Валентина

Санкт-Петербург
2020

Содержание

Цель.....	3
Задание.....	3
Иерархия.....	4
Что было добавлено.....	4
Выводы:.....	5
Код программы.....	5
Файл hat_circle.h.....	5
Файл circle.h.....	6
Файл shape.h.....	6
Файл face.h.....	7
Файл shape.cpp.....	7

Цель

Исследование принципов наследования и полиморфизма. Создание иерархии обетов

Задание

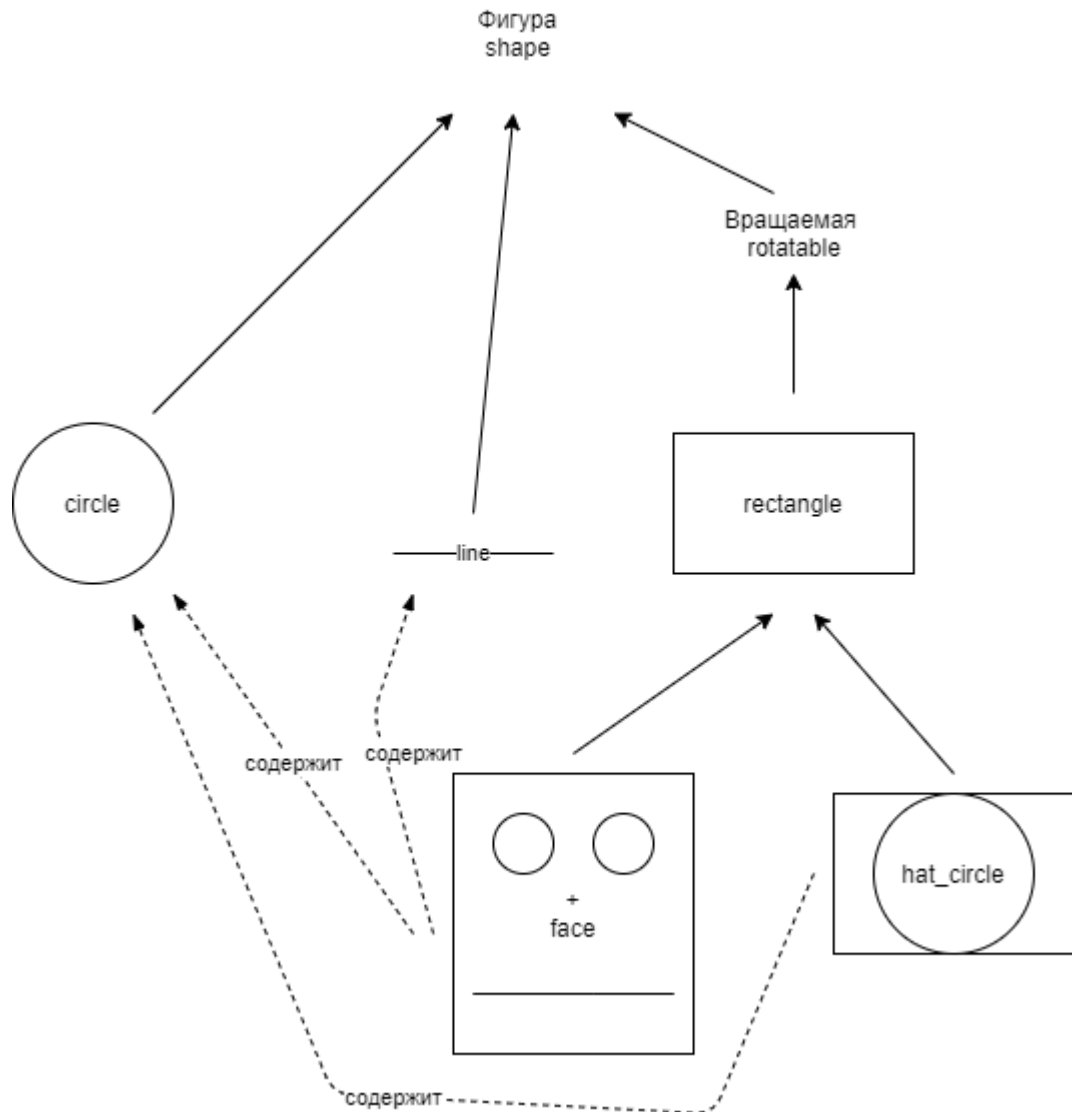
Доработать модуль *shape.cpp*, добавив в коллекцию кружок.

Расположение 10, 11, 12.



Для фигур, назначенных в позиции 6, 9, 10, 11 и 12, допускается замена отношения между классами ЯВЛЯЕТСЯ отношением СОДЕРЖИТ

Иерархия



Что было добавлено

Был добавлен класс circle. В нем были определены все функции-члены базового класса(shape), так как базовый класс является абстрактным. Добавлено приватное поле center, описывающее центр, и поле R, описывающее радиус.

Был добавлен класс face. В нем были переопределены следующие функции-члены:

- draw() для отрисовки носа
- move() для передвижения объектов, которые данный класс содержит, при его передвижении
- resize() для масштабирования объектов, содержащихся в данном классе, вместе с ним

Был добавлен класс hat_circle. В нем были переопределены следующие функции-члены:

- move() для передвижения объектов, содержащихся в данном классе, вместе с ним
- resize() для масштабирования объектов, которые данный класс содержит, при его масштабировании

У всех классов конструктор копирования, конструктор перемещения, присвоение, присвоение с перемещением были сделаны не доступными. Для этого был создан вспомогательный класс Uncopyable, в котором данные функции запрещены (=delete). Наш базовый для всех фигур класс shape был унаследован от класса Uncopyable, после этого компилятор не будет создавать не нужные нам функции-члены.

Выводы:

Наследование и полиморфизм помогают сократить количество кода, который нужно написать для добавления фигуры в библиотеку, так, например, для фигур лица и шляпы с эмблемой не были реализованы функции вычисления опорных точек, так как они реализованы в базовом классе.

Однако, если посмотреть на то, какие функции были нами все-таки переопределены, можно заметить, что они почти полностью пересекаются, и это не спроста, причина этому отсутствие системы иерархии у объектов непосредственно, что при передвижении или масштабировании объекта содержащиеся в нем никак не меняются и данное поведение приходится реализовывать вручную. Создав систему иерархии можно еще больше сократить количество кода, требующегося для добавления фигуры в библиотеку.

Код программы

Файл hat_circle.h

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

class hat_circle : public rectangle
{
    int w, h;
    circle circle;
public:
    hat_circle(point a, point b) : rectangle(a, b),
        w(neast().x - swest().x),
        h(neast().y - swest().y),
        circle(point(south().x, east().y), w >= h ? h / 2 : w / 2) {
        std::cout << east().y;
    }
    void resize(int d);
    void move(int, int);
};

void hat_circle::move(int a, int b)
{
    rectangle::move(a, b);
    circle.move(a, b);
}

void hat_circle::resize(int d)
{
    rectangle::resize(d);
    circle.resize(d);
    circle.move(south().x - circle.south().x, east().y - circle.east().y);
}
```

Файл circle.h

```
#include "../shape.h"

class circle : public shape
{
    point center;
    int R;
public:
    circle(point c, int r) : center(c), R(r) {}

    point north() const { return point(center.x, center.y + R); }
    point south() const { return point(center.x, center.y - R); }
    point east() const { return point(center.x + R, center.y); }
    point west() const { return point(center.x - R, center.y); }
    point neast() const { return point(center.x + R, center.y + R); }
    point seast() const { return point(center.x + R, center.y - R); }
    point nwest() const { return point(center.x - R, center.y + R); }
    point swest() const { return point(center.x - R, center.y - R); }
    void draw(); //Рисование
    void move(int, int); //Перемещение
    void resize(int); //Изменение размера

    int radius()
    {
        return R;
    }
};

void circle::draw() //Алгоритм Брезенхэма для окружностей
{
    //(выдаются два сектора, указываемые значением reflected)
    int x0 = center.x, y0 = center.y;
    int radius = R;
    int x = 0, y = radius, delta = 1 - 2 * radius, error = 0;
    float k = 1;
    while (y >= 0)
    { // цикл рисования

        put_point(x0 + x, y0 + y * k);
        put_point(x0 + x, y0 - y * k);
        put_point(x0 - x, y0 + y * k);
        put_point(x0 - x, y0 - y * k);

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

void circle::resize(int d)
{
    R *= d;
}

void circle::move(int a, int b)
{
    center.x += a;
    center.y += b;
}
```

Файл shape.h

```
#include "screen.h"
#include <list>

class Uncopyable
{
protected:
    Uncopyable() = default;
    ~Uncopyable() = default;
```

```

    Uncopyable(const Uncopyable&)=delete;
    Uncopyable(const Uncopyable&&) = delete;
    Uncopyable& operator=(const Uncopyable&) = delete;
    Uncopyable& operator=(const Uncopyable&&) = delete;
};

//== 2. Библиотека фигур ==
struct shape : private Uncopyable
{
    ...
}

```

Файл face.h

```

#include "rectangle.h"
#include "circle.h"
#include "line.h"

class face : public rectangle { // моя фигура является
    int w, h; // прямоугольником
    circle l_eye; // левый глаз
    circle r_eye; // правый глаз
    line mouth; // рот
public:
    face(point, point);
    void draw();
    void move(int, int);
    void resize(float d)
    {
        w *= d;
        h *= d;
        rectangle::resize(d);
        l_eye.resize(2 * d / 3);
        l_eye.move(swest().x + 2 * d - l_eye.swest().x, neast().y -
l_eye.radius() - 2 * d - l_eye.swest().y);
        r_eye.resize(2 * d / 3);
        r_eye.move(neast().x - r_eye.radius() * 2 - 2 * d - r_eye.swest().x,
neast().y - r_eye.radius() - 2 * d - r_eye.swest().y);
        mouth.resize(d);
        mouth.move(swest().x - mouth.swest().x + 2 * d, swest().y -
mouth.swest().y + h / 4);
    }
};

face::face(point a, point b)
: rectangle(a, b), //инициализация базового класса
w(neast().x - swest().x + 1), // инициализация данных
h(neast().y - swest().y + 1), // - строго в порядке объявления!
l_eye(point(swest().x + 3, neast().y - 1 - 1), 1),
r_eye(point(neast().x - 1 * 2 - 1, neast().y - 1 - 1), 1),
mouth(point(swest().x + 2, swest().y + h / 4), w - 4)
{ }

void face::draw()
{
    rectangle::draw(); //контур лица (глаза и нос рисуются сами!)
    int a = (swest().x + neast().x) / 2;
    int b = (swest().y + neast().y) / 2;
    put_point(point(a, b)); // Нос - существует только на рисунке!
}

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);
}

```

Файл shape.cpp

```

#define _WIN32_WINNT 0x0500
//it is important that the above line be typed
// BEFORE <windows.h> is included
#include <windows.h>

#include "screen.h"
#include "shape.h"
#include "shapes/hat_circle.h"
#include "shapes/line.h"
#include "shapes/face.h"

```

```

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);
}
// Сборная пользовательская фигура - физиономия

void up(shape& p, const shape& q) // поместить p над q
{
    //Это ОБЫЧНАЯ функция, не член класса! Динамическое связывание!!
    point n = q.north();
    point s = p.south();
    p.move(n.x - s.x, n.y - s.y + 1);
}

int main()
{
    HWND console = GetConsoleWindow();
    RECT r;
    GetWindowRect(console, &r); //stores the console's current dimensions

    //Movewindow(window_handle, x, y, width, height, redraw_window);
    Movewindow(console, r.left, r.top, XMAX * 10, YMAX * 10, TRUE);

    setlocale(LC_ALL, "Rus");
    screen_init();
    //== 1.Объявление набора фигур ==
    hat_circle hat(point(1, 1), point(15, 7));
    line brim(point(0, 15), 17);
    face face(point(15, 10), point(27, 18));
    shape_refresh();
    std::cout << "=== Generated... ===\n";
    std::cin.get(); //Смотреть исходный набор

    //== 2.Подготовка к сборке ==
    hat.rotate_right();
    hat.resize(3);
    brim.resize(3);
    face.resize(3);
    face.move(-8, -9);
    shape_refresh();
    std::cout << "=== Prepared... ===\n";
    std::cin.get(); //Смотреть результат поворотов/отражений

    //== 3.Сборка изображения ==
    //face.move(0, -10); // лицо - в исходное положение
    up(brim, face);
    up(hat, brim);
    shape_refresh();
    std::cout << "=== Ready! ===\n";
    std::cin.get(); //Смотреть результат
    screen_destroy();
    return 0;
}

```