**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. 7307 | Шалугин Е.Д. Васильев А.В. |  |
| Преподаватель | Колинько П.Г. |  |

Санкт-Петербург

2019

Оглавление

[Цель работы 3](#__RefHeading___Toc188_4150899979)

[Задание 3](#__RefHeading___Toc190_4150899979)

[Иерархия классов 3](#__RefHeading___Toc192_4150899979)

[Добавленный код 4](#__RefHeading___Toc194_4150899979)

[Вывод 7](#__RefHeading___Toc196_4150899979)

# **Цель работы**

Изучить наследование и полиморфизм на примере программы рисующей различные фигуры.

# **Задание**

Доработать модуль shape.cpp: добавить в коллекцию ещё одну фигуру — трапецию. Для фигуры определить подходящее место в иерархии классов и написать необходимые функции - члены. Разработанной фигурой нужно дополнить картинку в позициях 1, 4 и 5 (галстук и уши), которые должны быть повернуты или отражены.

# **Иерархия классов**

# C:\Users\mi\Downloads\Untitled Diagram (18).jpg

# **Добавленный код**

1. class trapezioid : public rotatable, public reflectable {
2. /\* nw ------ n ------ ne
3. / \
4. / \
5. e c w
6. / \
7. / \
8. sw ----------- s ----------- se \*/
9. protected:
10. point sw, se, nw, ne;
11. public:
12. trapezioid(point a, point b, point c, point d) : sw(a), se(b), nw(c), ne(d)
13. {
14. try
15. {
16. if (b.x > XMAX || a.x < 0 ||
17. c.y > YMAX || a.y < 0 ||
18. d.x > XMAX || c.x < 0) {
19. flag = true;
20. throw ShapeOutOfScreenException();
21. }
22. else
23. if (a.y - c.y != b.y - d.y || a.x > b.x || c.x > d.x || c.y < a.y) {
24. flag = true;
25. throw WrongShapeException();
26. }
27. //sw = a;
28. //ne = b;
29. }
30. catch (ShapeOutOfScreenException)
31. {
32. cout << "Trapezioid (" << a.x << ", " << a.y << ") - (" << b.x << ", " << b.y << ") is out of screen!" << endl;
33. //sw = point::zero();
34. //ne = point::zero();
35. }
36. catch (WrongShapeException)
37. {
38. cout << "Trapezioid (" << a.x << ", " << a.y << ") - (" << b.x << ", " << b.y << ") adjusted!" << endl;
39. //sw.x = a.x < b.x ? a.x : b.x;
40. //sw.y = a.y < b.y ? a.y : b.y;
41. //ne.x = a.x > b.x ? a.x : b.x;
42. //ne.y = a.y > b.y ? a.y : b.y;
43. }
44. }
45. //trapezioid(point a, point b, point c, point d) : sw(a), se(b), nw(c), ne(d) { }
46. point north() const { return point((nw.x + ne.x) / 2, ne.y); }
47. point south() const { return point((sw.x + se.x) / 2, sw.y); }
48. point east() const { return point((ne.x + se.x) / 2, (se.y + ne.y) / 2); }
49. point west() const { return point((sw.x + nw.x) / 2, (sw.y + nw.y) / 2); }
50. point neast() const { return ne; }
51. point seast() const { return se; }
52. point nwest() const { return nw; }
53. point swest() const { return sw; }
54. void rotate\_right() // Поворот вправо относительно se
55. {
56. int l1 = nw.x - sw.x;
57. int h = nw.y - sw.y;
58. int l2 = se.x - ne.x;
59. se.y = sw.y - (se.x - sw.x);
60. se.x = sw.x;
61. nw.x = se.x + h;
62. nw.y = sw.y - l1;
63. ne.x = nw.x;
64. ne.x = nw.x;
65. ne.y = se.y + l2;
66. //int w = ne.x - sw.x, h = ne.y - sw.y;
67. //sw.x = ne.x - h \* 2; ne.y = sw.y + w / 2;
68. }
69. void rotate\_left() // Поворот влево относительно sw
70. {
71. int l1 = nw.x - sw.x;
72. int h = nw.y - sw.y;
73. int l2 = se.x - ne.x;
74. se.y = sw.y + (se.x - sw.x);
75. se.x = sw.x;
76. nw.x = se.x - h;
77. nw.y = sw.y + l1;
78. ne.x = nw.x;
79. ne.y = se.y - l2;
80. //int w = ne.x - sw.x, h = ne.y - sw.y;
81. //ne.x = sw.x + h \* 2; ne.y = sw.y + w / 2;
82. }
83. void move(int a, int b)
84. {
85. sw.x += a; sw.y += b; ne.x += a; ne.y += b;
86. nw.x += a; nw.y += b; se.x += a; se.y += b;
87. }
88. void flip\_horisontally() {
89. int p1 = nw.y;
90. nw.y = se.y;
91. ne.y = se.y;
92. se.y = p1;
93. sw.y = p1;
94. }
95. void flip\_vertically() {
96. int p1 = nw.x;
97. nw.x = ne.x;
98. ne.x = p1;
99. p1 = sw.x;
100. sw.x = se.x;
101. se.x = p1;
102. }
103. void resize(int blabla){
104. }
105. void draw();
106. };
107. void trapezioid::draw()
108. {
109. put\_line(nwest(), ne); put\_line(ne, seast());
110. put\_line(seast(), sw); put\_line(sw, nwest());
111. }
112. void up(shape& p, const shape& q) // поместить p над q
113. { //Это ОБЫЧНАЯ функция, а не член класса!
114. point n = q.north();
115. point s = p.south();
116. p.move(n.x - s.x, n.y - s.y + 1);
117. }
118. void left(shape& p, const shape& q) // поместить p над q
119. { //Это ОБЫЧНАЯ функция, а не член класса!
120. point e = q.west();
121. point s = p.south();
122. p.move(e.x - s.x, e.y - s.y - 4);
123. }
124. void right(shape& p, const shape& q) // поместить p над q
125. { //Это ОБЫЧНАЯ функция, а не член класса!
126. point n = q.east();
127. point s = p.south();
128. p.move(n.x - s.x, n.y - s.y + 4);
129. }

# **Результаты работы программы**

# 

# **Вывод**

В ходе работы мы ознакомились с понятиями наследование и полиморфизм. Благодаря наследованию удалось сократить количество кода при написании новых фигур. Мы не использовали наследование трапеции от прямоугольника для большей функциональности при работе с фигурой и простоте работы с функциями.