Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 9

по курсу «Языки и методы программирования»

«Перегрузка операций»

Студент группы ИУ9-22Б Федуков А. А.

Преподаватель Посевин Д. П.

8 мая 2024 г.

Цель работы

Данная работа предназначена для изучения возможностей языка C++, обеспечивающих применение знаков операций к объектам пользовательских типов.

Задание

Согласно выбранному из таблиц 1–26 описанию требуется составить шаблон класса, перегрузив указанные операции.

Задание 1

Shape<T> – множество точек в пространстве T × T, задающих некоторую геометрическую фигуру. Операции, перегружаемые для Shape<T>:

1. «+» и «-» - объединение и разность двух множеств; 2. «( )» – проверка принадлежности точки множеству.

У класса Shape<T> должно быть два конструктора: 1. первый конструктор принимает координаты нижней левой и верхней правой вершин прямоугольника, каждая сторона которого параллельна одной из осей координат, и порождает множество точек, принадлежащих этому прямоугольнику; 2. аналогично, второй конструктор порождает множество точек круга по координатам центра и радиусу.

Реализация

Я описал класс и проверил работоспособность его необходимых функций уже в main.cpp

Код

Листинг 1: Файл main.cpp

```
1 #include <iostream>
2 #include <vector>
3 #include <cmath>
```

```
4  #include <stdexcept>
5
6  #define ERROR throw std::invalid_argument("Outside of the area!")
7  #define CHECK_POINT(x, y, T) (x > T || y > T || x < 0 || y < 0) ? 1 : 0
8
9  template <int T>
10 class Shape_
11 {
12 private:
13     int type; // 1 - rectangle  ; 2 - circle
14     int x1, y1, x2, y2;
15     int x, y, r;
16
17 public:
18     Shape_(int x1, int y1, int x2, int y2); // rect
19     Shape_(int x, int y, int r);            // circle
20     bool Inhere(int x, int y);
21 };
22
23 template <int T>
24 Shape_<T>::Shape_(int x1, int y1, int x2, int y2)
25 {
26     if (CHECK_POINT(x1, y1, T) || CHECK_POINT(x2, y2, T))
27         ERROR;
28
29     this->x1 = x1;
30     this->y1 = y1;
31     this->x2 = x2;
32     this->y2 = y2;
33     this->type = 1;
34 }
35
36 template <int T>
37 Shape_<T>::Shape_(int x, int y, int r)
38 {
39     if (CHECK_POINT(x, y, T) || r > T || r < 0)
40         ERROR;
41
42     this->x = x;
43     this->y = y;
44     this->r = r;
45     this->type = 2;
46 }
47
48 template <int T>
49 bool Shape_<T>::Inhere(int x, int y)
```

```
50 {
51
52     if (type == 1)
53     {
54         return x >= x1 && y >= y1 && x <= x2 && y <= y2;
55     }
56     else if (type == 2)
57     {
58         return pow((x - this->x), 2) + pow((y - this->y), 2) <= pow(r,
       2);
59     }
60     return false;
61 }
62
63 template <int T>
64 class Shape
65 {
66 private:
67     std::vector<Shape_<T>> shapesPlus;
68     std::vector<Shape_<T>> shapesMinus;
69
70 public:
71     Shape(int x1, int y1, int x2, int y2); // rect
72     Shape(int x, int y, int r);            // circle
73     Shape<T> operator+(const Shape<T> &shape);
74     Shape<T> operator-(const Shape<T> &shape);
75     bool operator()(int x, int y);
76 };
77
78 template <int T>
79 Shape<T>::Shape(int x1, int y1, int x2, int y2)
80 {
81     shapesPlus.push_back(Shape_<T>(x1, y1, x2, y2));
82 }
83
84 template <int T>
85 Shape<T>::Shape(int x, int y, int r)
86 {
87     shapesPlus.push_back(Shape_<T>(x, y, r));
88 }
89
90 template <int T>
91 Shape<T> Shape<T>::operator+(const Shape<T> &shape)
92 {
93     for (auto &&i : shape.shapesPlus)
94         this->shapesPlus.push_back(i);
```

```
95
96      for (auto &&i : shape.shapesMinus)
97          this->shapesMinus.push_back(i);
98
99      return *this;
100 }
101
102 template <int T>
103 Shape<T> Shape<T>::operator-(const Shape<T> &shape)
104 {
105     for (auto &&i : shape.shapesPlus)
106         this->shapesMinus.push_back(i);
107
108     for (auto &&i : shape.shapesMinus)
109         this->shapesPlus.push_back(i);
110
111     return *this;
112 }
113
114 template <int T>
115 bool Shape<T>::operator()(int x, int y)
116 {
117     if (CHECK_POINT(x, y, T))
118         ERROR;
119
120     int inside = 0;
121
122     for (auto &&i : shapesPlus)
123         if (i.Inhere(x, y))
124             inside++;
125
126     for (auto &&i : shapesMinus)
127         if (i.Inhere(x, y))
128             inside--;
129
130     return inside > 0;
131 }
132
133 int main()
134 {
135     Shape<100> cc{10, 10, 5};
136     // Возможен ли мудрый принт объектов, как перегрузка toString в Java
    ?
137     std::cout << "Check (7, 8) of {10, 10, 5}: " << cc(7, 8) << std::
    endl;
```

```
138     std::cout << "Check (7, 8) minus area: " << (cc - Shape<100>(0, 0,
        100, 100))(7, 8) << std::endl;
139     std::cout << "Check (7, 8) plus area: " << (cc + Shape<100>(9, 9,
        10))(7, 8) << std::endl;
140     return 0;
141 }
```

Вывод программы

Программа создала экземляры класса и протестировала все заявленные операции

Листинг 2: Вывод программы

```
1 Check (7, 8) of {10, 10, 5}: 1
2 Check (7, 8) minus area: 0
3 Check (7, 8) plus area: 1
```

Вывод

Выполняя эту лабораторную работу, я научился перегрузке операций для объектов классов в C++.