

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

старший преподаватель		Е.О. Шумова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Перегрузка операторов

по дисциплине: Объектно-ориентированное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	Z1431		М.Д. Быстров
	номер группы	подпись, дата	инициалы, фамилия

Студенческий билет №	2021/3572
----------------------	-----------

Санкт-Петербург 2023

## **Условие**

Цель работы: изучить механизм перегрузки операторов для типов, определенных пользователем посредством использования методов класса и дружественных функций.

Закрепить знания по теме: перегрузка операторов.

Описание работы: В работе необходимо реализовать класс в соответствии с вариантом задания и создать приложение. В классе должен быть предусмотрен конструктор для установки начальных значений полей, методы – члены класса и дружественные методы, обеспечивающие перегрузку операций для заданного класса. Часть перегруженных операторов должны быть членами класса, а часть – дружественными функциями. Т.е. в работе должны быть продемонстрированы оба способа перегрузки операторов (не нужно один и тот же оператор перегружать двумя способами).

Индивидуальное задание (вариант 2):

2. Разработать класс «Прямоугольник», в котором содержатся поля для хранения 4 вершин прямоугольника. Определить в нем конструкторы и деструктор, перегрузить операцию объединения прямоугольников (операция “\*”) для случая успешного выполнения перегруженной операции проверки совпадения сторон с равной длиной), операцию вычисления периметра прямоугольника, операции сравнения двух прямоугольников (по периметру).

# Полный текст (листинг) программы

## 1. Файл «main.cpp»

```
1. #include <iostream>
2. #include <string>
3. #include <Windows.h>
4. #include "Polygon.h"
5.
6. #define FIRST_POLYGON_COORDS 1,1,2,1,2,2,1,2
7. #define SECOND_POLYGON_COORDS 2,1,3,1,3,2,2,2
8.
9. using namespace std;
10.
11. int main()
12. {
13.     SetConsoleCP(1251);
14.     SetConsoleOutputCP(1251);
15.
16.     cout << "JP №3 ВАРИАНТ 2" << endl << endl;
17.
18.     MyPolygon* polygon1 = new MyPolygon(FIRST_POLYGON_COORDS);
19.     MyPolygon* polygon2 = new MyPolygon(SECOND_POLYGON_COORDS);
20.
21.     MyPolygon* polygon3 = &((*polygon1) * (*polygon2));
22.
23.     cout << polygon1->getDescription() << " * "
24.         << polygon2->getDescription() << " is "
25.         << polygon3->getDescription() << endl << endl;
26.
27.     cout << "Perimeter of " << polygon1->getDescription() << " is " <<
        polygon1->getPerimeter() << endl;
28.     cout << "Perimeter of " << polygon2->getDescription() << " is " <<
        polygon2->getPerimeter() << endl;
29.     cout << "Perimeter of " << polygon3->getDescription() << " is " <<
        polygon3->getPerimeter() << endl;
30.
31.     cout << endl;
32.
33.     bool first_equal = (*polygon1) == (*polygon2);
34.     bool second_equal = (*polygon1) == (*polygon3);
35.
36.     cout << polygon1->getDescription() << " is equal to "
37.         << polygon2->getDescription() << " : "
38.         << first_equal << endl;
39.
40.     cout << polygon1->getDescription() << " is equal to "
```

```

41.         << polygon3->getDescription() << " : "
42.         << second_equal << endl;
43.
44.     cout << endl;
45.
46.     delete polygon1;
47.     delete polygon2;
48.     delete polygon3;
49. }

```

## 2. Файл «Polygon.h»

```

1. #include <string>
2.
3. using namespace std;
4.
5. class MyPolygon
6. {
7. private:
8.
9.     int x1;
10.    int y1;
11.
12.    int x2;
13.    int y2;
14.
15.    int x3;
16.    int y3;
17.
18.    int x4;
19.    int y4;
20.
21. public:
22.
23.    MyPolygon(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int
        y4);
24.
25.    MyPolygon& operator *(const MyPolygon& right_polygon);
26.
27.    friend bool operator ==(const MyPolygon & left_polygon, const
        MyPolygon & right_polygon);
28.
29.    bool hasEqualSide(const MyPolygon& right_polygon);
30.
31.    int getX1() const;
32.    void setX1(int x1);
33.

```

```

34.     int getY1() const;
35.     void setY1(int y1);
36.
37.     int getX2() const;
38.     void setX2(int x2);
39.
40.     int getY2() const;
41.     void setY2(int y2);
42.
43.     int getX3() const;
44.     void setX3(int x3);
45.
46.     int getY3() const;
47.     void setY3(int y3);
48.
49.     int getX4() const;
50.     void setX4(int x4);
51.
52.     int getY4() const;
53.     void setY4(int y4);
54.
55.     string getDescription() const;
56.
57.     int getPerimeter() const;
58.
59.     ~MyPolygon();
60. };

```

### 3. Файл «Polygon.cpp»

```

1. #include <iostream>
2. #include <sstream>
3. #include "Polygon.h"
4.
5. MyPolygon::MyPolygon(int x1, int y1, int x2, int y2, int x3, int y3, int
   x4, int y4)
6. {
7.     if (x1 != x4
8.         || x2 != x3
9.         || y1 != y2
10.        || y3 != y4
11.        || x2 <= x1
12.        || y3 <= y2)
13.     {
14.         throw std::invalid_argument("Wrong polygon coordinates");
15.     }

```

```

16.
17.     this->x1 = x1;
18.     this->x2 = x2;
19.     this->x3 = x3;
20.     this->x4 = x4;
21.
22.     this->y1 = y1;
23.     this->y2 = y2;
24.     this->y3 = y3;
25.     this->y4 = y4;
26. }
27.
28. MyPolygon& MyPolygon::operator*(const MyPolygon& right_polygon)
29. {
30.     if (!(this->hasEqualSide(right_polygon)))
31.     {
32.         throw std::invalid_argument("Polygons has no equal sides");
33.     }
34.
35.     int newX1 = this->getX1() < right_polygon.x1 ? this->getX1() :
right_polygon.x1;
36.     int newY1 = this->getY1() < right_polygon.y1 ? this->getY1() :
right_polygon.y1;
37.
38.     int newX2 = this->getX2() > right_polygon.x2 ? this->getX2() :
right_polygon.x2;
39.     int newY2 = this->getY2() < right_polygon.x2 ? this->getY2() :
right_polygon.y2;
40.
41.     int newX3 = this->getX3() > right_polygon.x3 ? this->getX3() :
right_polygon.x3;
42.     int newY3 = this->getY3() > right_polygon.x3 ? this->getY3() :
right_polygon.y3;
43.
44.     int newX4 = this->getX4() < right_polygon.x4 ? this->getX4() :
right_polygon.x4;
45.     int newY4 = this->getY4() > right_polygon.x4 ? this->getY4() :
right_polygon.y4;
46.
47.     MyPolygon* polygon = new MyPolygon(newX1, newY1, newX2, newY2, newX2,
newY3, newX4, newY4);
48.
49.     return *polygon;
50. }
51.
52. MyPolygon::~MyPolygon()
53. {
54.     std::cout << "Destruction of "
55.         << this->getDescription()
56.         << std::endl;

```

```

57.}
58.
59.string MyPolygon::getDescription() const
60.{
61.    std::stringstream buffer;
62.
63.    buffer << "Polygon "
64.        << "[" << x1 << ";" << y1 << "], "
65.        << "[" << x2 << ";" << y2 << "], "
66.        << "[" << x3 << ";" << y3 << "], "
67.        << "[" << x4 << ";" << y4 << "];"
68.
69.    return buffer.str();
70.}
71.
72.int MyPolygon::getPerimeter() const
73.{
74.    const int coordNum = 8;
75.
76.    int polygonTops[coordNum] = {
77.        this->getX1(),
78.        this->getY1(),
79.        this->getX2(),
80.        this->getY2(),
81.        this->getX3(),
82.        this->getY3(),
83.        this->getX4(),
84.        this->getY4() };
85.
86.    int perimeter = 0;
87.
88.    for (int i = 0; i < coordNum; i += 2)
89.    {
90.        int x1 = polygonTops[i];
91.        int y1 = polygonTops[i + 1];
92.        int x2 = polygonTops[(i + 2) % coordNum];
93.        int y2 = polygonTops[(i + 3) % coordNum];
94.
95.        if (x1 != x2)
96.        {
97.            perimeter += x1 > x2 ? x1 - x2 : x2 - x1;
98.        }
99.        else
100.        {
101.            perimeter += y1 > y2 ? y1 - y2 : y2 - y1;
102.        }
103.    }
104.
105.    return perimeter;
106.}

```

```

107.
108.     int MyPolygon::getX1() const { return this->x1; }
109.     void MyPolygon::setX1(int x1) { this->x1 = x1; }
110.     int MyPolygon::getY1() const { return this->y1; }
111.     void MyPolygon::setY1(int y1) { this->y1 = y1; }
112.
113.     int MyPolygon::getX2() const { return this->x2; }
114.     void MyPolygon::setX2(int x2) { this->x2 = x2; }
115.     int MyPolygon::getY2() const { return this->y2; }
116.     void MyPolygon::setY2(int y2) { this->y2 = y2; }
117.
118.     int MyPolygon::getX3() const { return this->x3; }
119.     void MyPolygon::setX3(int x3) { this->x3 = x3; }
120.     int MyPolygon::getY3() const { return this->y3; }
121.     void MyPolygon::setY3(int y3) { this->y3 = y3; }
122.
123.     int MyPolygon::getX4() const { return this->x4; }
124.     void MyPolygon::setX4(int x4) { this->x4 = x4; }
125.     int MyPolygon::getY4() const { return this->y4; }
126.     void MyPolygon::setY4(int y4) { this->y4 = y4; }
127.
128.     bool MyPolygon::hasEqualSide(const MyPolygon& right_polygon)
129.     {
130.         const int coordNum = 8;
131.
132.         int leftPolygonTops[coordNum] = {
133.             this->getX1(),
134.             this->getY1(),
135.             this->getX2(),
136.             this->getY2(),
137.             this->getX3(),
138.             this->getY3(),
139.             this->getX4(),
140.             this->getY4() };
141.
142.         int rightPolygonTops[coordNum] = {
143.             right_polygon.getX1(),
144.             right_polygon.getY1(),
145.             right_polygon.getX2(),
146.             right_polygon.getY2(),
147.             right_polygon.getX3(),
148.             right_polygon.getY3(),
149.             right_polygon.getX4(),
150.             right_polygon.getY4() };
151.
152.         for (int i = 0; i < coordNum; i += 2)
153.         {
154.             int leftX1 = leftPolygonTops[i];
155.             int leftY1 = leftPolygonTops[i + 1];
156.             int leftX2 = leftPolygonTops[(i + 2) % coordNum];

```

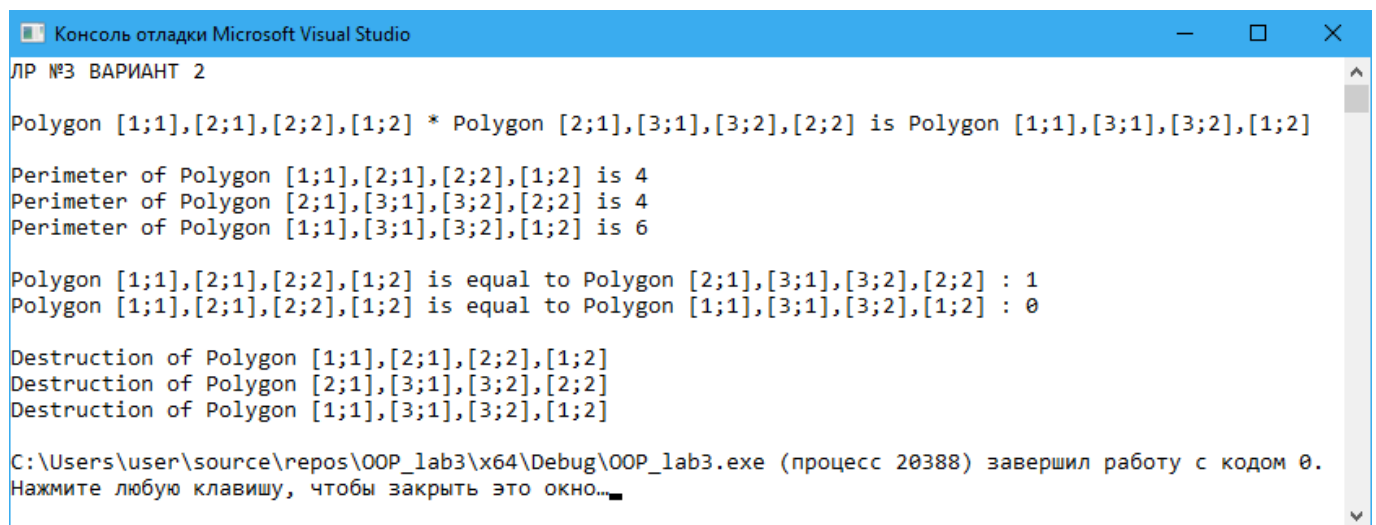


```

157.         int leftY2 = leftPolygonTops[(i + 3) % coordNum];
158.
159.         for (int j = 0; j < coordNum; j += 2)
160.         {
161.             int rightX1 = rightPolygonTops[j];
162.             int rightY1 = rightPolygonTops[j + 1];
163.             int rightX2 = rightPolygonTops[(j + 2) % coordNum];
164.             int rightY2 = rightPolygonTops[(j + 3) % coordNum];
165.
166.             if ((leftX1 == rightX1
167.                 && leftY1 == rightY1
168.                 && leftX2 == rightX2
169.                 && leftY2 == rightY2)
170.                 || (leftX1 == rightX2
171.                     && leftY1 == rightY2
172.                     && leftX2 == rightX1
173.                     && leftY2 == rightY1))
174.             {
175.                 return true;
176.             }
177.         }
178.     }
179.
180.     return false;
181. }
182.
183. bool operator==(const MyPolygon& left_polygon, const MyPolygon&
    right_polygon)
184. {
185.     return left_polygon.getPerimeter() ==
        right_polygon.getPerimeter();
186. }
187.

```

## Работа программы



Консоль отладки Microsoft Visual Studio

ЛР №3 ВАРИАНТ 2

Polygon [1;1],[2;1],[2;2],[1;2] \* Polygon [2;1],[3;1],[3;2],[2;2] is Polygon [1;1],[3;1],[3;2],[1;2]

Perimeter of Polygon [1;1],[2;1],[2;2],[1;2] is 4

Perimeter of Polygon [2;1],[3;1],[3;2],[2;2] is 4

Perimeter of Polygon [1;1],[3;1],[3;2],[1;2] is 6

Polygon [1;1],[2;1],[2;2],[1;2] is equal to Polygon [2;1],[3;1],[3;2],[2;2] : 1

Polygon [1;1],[2;1],[2;2],[1;2] is equal to Polygon [1;1],[3;1],[3;2],[1;2] : 0

Destruction of Polygon [1;1],[2;1],[2;2],[1;2]

Destruction of Polygon [2;1],[3;1],[3;2],[2;2]

Destruction of Polygon [1;1],[3;1],[3;2],[1;2]

C:\Users\user\source\repos\OOP\_lab3\x64\Debug\OOP\_lab3.exe (процесс 20388) завершил работу с кодом 0.  
Нажмите любую клавишу, чтобы закрыть это окно...

*Рисунок 1 Работа программы*

## **Выводы**

В ходе выполнения лабораторной работы №3 были получены навыки по перегрузке операторов и работе с дружественными функциями класса.

Переопределены операторы в соответствии с вариантом. Один оператор переопределен с помощью метода-члена класса, другой оператор переопределен с помощью дружественной функции. Таким образом, продемонстрированы разные способы перегрузки операторов в C++.