ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «Перегрузка операторов» |
| по курсу: ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель работы**

Изучить механизм перегрузки операторов для типов, определенных пользователем посредством использования методов класса и дружественных функций.

**Вариант 3**

3. Разработать класс «Треугольник», содержащий поля для хранения 3 вершин треугольника. Определить в нем конструкторы и деструктор, перегрузить операцию проверки существования треугольника, операцию проверки включения точки в треугольник, операции сравнения двух треугольников (по площади)

**Листинг**

#include <iostream>

#include <math.h>

class Triangle

{

public:

Triangle(double vxa = 0, double vya = 0, double vxb = 3, double vyb = 0, double vxc = 0, double vyc = 4);

~Triangle();

bool operator~();

double \*get\_a();

double \*get\_b();

double \*get\_c();

bool operator+ (double arr[2]);

friend bool operator<(Triangle obj, Triangle right);

friend bool operator>(Triangle obj, Triangle right);

friend bool operator<=(Triangle obj, Triangle right);

friend bool operator<=(Triangle obj, Triangle right);

private:

double xa, xb, xc, ya, yb, yc;

};

Triangle::Triangle(double vxa, double vya , double vxb , double vyb , double vxc , double vyc )

{

xa = vxa;

ya = vya;

xb = vxb;

yb = vyb;

xc = vxc;

yc = vyc;

}

Triangle::~Triangle()

{

}

double \*Triangle::get\_a()

{

double point[2] = { xa, ya };

return point;

}

double\* Triangle::get\_b()

{

double point[2] = { xb, yb };

return point;

}

double\* Triangle::get\_c()

{

double point[2] = { xc, yc };

return point;

}

bool Triangle::operator~()

{

double ab, bc, ac;

double x1, x2, x3, y1, y2, y3;

x1 = this->get\_a()[0];

y1 = this->get\_a()[1];

x2 = this->get\_b()[0];

y2 = this->get\_b()[1];

x3 = this->get\_c()[0];

y3 = this->get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

//std::cout << ab << " " << ac << " " << bc << std:: endl;

if (ab < (bc + ac) && bc < (ac + ab) && ac < (bc + ab))

{

return 1;

}

else

{

return 0;

}

}

bool Triangle::operator+(double arr[2])

{

double x1, x2, x3, y1, y2, y3, x0, y0;

x1 = this->get\_a()[0];

y1 = this->get\_a()[1];

x2 = this->get\_b()[0];

y2 = this->get\_b()[1];

x3 = this->get\_c()[0];

y3 = this->get\_c()[1];

x0 = arr[0];

y0 = arr[1];

if ((((x1 - x0) \* (y2 - y1) - (x2 - x1) \* (y1 - y0) <= 0) && ((x2 - x0) \* (y3 - y2) - (x3 \* x2) \* ( y2-y0)<=0) && ((x3-x0)\*(y1-y3)-(x1-x3)\*(y3-y0)<=0)) || (((((x1 - x0) \* (y2 - y1) - (x2 - x1) \* (y1 - y0) >= 0) &&((x2 - x0) \* (y3 - y2) - (x3 \* x2) \* (y2 - y0) >=0) && ((x3 - x0) \* (y1 - y3) - (x1 - x3) \* (y3 - y0) >= 0)))))

{

return 1;

}

else

{

return 0;

}

}

bool operator>(Triangle obj, Triangle right)

{

double ab, bc, ac;

double x1, x2, x3, y1, y2, y3;

x1 = obj.get\_a()[0];

y1 = obj.get\_a()[1];

x2 = obj.get\_b()[0];

y2 = obj.get\_b()[1];

x3 = obj.get\_c()[0];

y3 = obj.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p = (ab + ac + bc) / 2;

double s = sqrt(p \* (p - ab) \* (p - bc) \* (p - ac));

x1 = right.get\_a()[0];

y1 = right.get\_a()[1];

x2 = right.get\_b()[0];

y2 = right.get\_b()[1];

x3 = right.get\_c()[0];

y3 = right.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p1 = (ab + ac + bc) / 2;

double s1 = sqrt(p1 \* (p1 - ab) \* (p1 - bc) \* (p1 - ac));

//std::cout << s << " " << s1 << std::endl;

if (s > s1)

{

return 1;

}

else

{

return 0;

}

}

bool operator>=(Triangle obj, Triangle right)

{

double ab, bc, ac;

double x1, x2, x3, y1, y2, y3;

x1 = obj.get\_a()[0];

y1 = obj.get\_a()[1];

x2 = obj.get\_b()[0];

y2 = obj.get\_b()[1];

x3 = obj.get\_c()[0];

y3 = obj.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p = (ab + ac + bc) / 2;

double s = sqrt(p \* (p - ab) \* (p - bc) \* (p - ac));

x1 = right.get\_a()[0];

y1 = right.get\_a()[1];

x2 = right.get\_b()[0];

y2 = right.get\_b()[1];

x3 = right.get\_c()[0];

y3 = right.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p1 = (ab + ac + bc) / 2;

double s1 = sqrt(p1 \* (p1 - ab) \* (p1 - bc) \* (p1 - ac));

//std::cout << s << " " << s1 << std::endl;

if (s >= s1)

{

return 1;

}

else

{

return 0;

}

}

bool operator<(Triangle obj, Triangle right)

{

double ab, bc, ac;

double x1, x2, x3, y1, y2, y3;

x1 = obj.get\_a()[0];

y1 = obj.get\_a()[1];

x2 = obj.get\_b()[0];

y2 = obj.get\_b()[1];

x3 = obj.get\_c()[0];

y3 = obj.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p = (ab + ac + bc) / 2;

double s = sqrt(p \* (p - ab) \* (p - bc) \* (p - ac));

x1 = right.get\_a()[0];

y1 = right.get\_a()[1];

x2 = right.get\_b()[0];

y2 = right.get\_b()[1];

x3 = right.get\_c()[0];

y3 = right.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p1 = (ab + ac + bc) / 2;

double s1 = sqrt(p1 \* (p1 - ab) \* (p1 - bc) \* (p1 - ac));

//std::cout << s << " " << s1 << std::endl;

if (s < s1)

{

return 1;

}

else

{

return 0;

}

}

bool operator<=(Triangle obj, Triangle right)

{

double ab, bc, ac;

double x1, x2, x3, y1, y2, y3;

x1 = obj.get\_a()[0];

y1 = obj.get\_a()[1];

x2 = obj.get\_b()[0];

y2 = obj.get\_b()[1];

x3 = obj.get\_c()[0];

y3 = obj.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p = (ab + ac + bc) / 2;

double s = sqrt(p \* (p - ab) \* (p - bc) \* (p - ac));

x1 = right.get\_a()[0];

y1 = right.get\_a()[1];

x2 = right.get\_b()[0];

y2 = right.get\_b()[1];

x3 = right.get\_c()[0];

y3 = right.get\_c()[1];

ab = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

bc = sqrt((x3 - x2) \* (x3 - x2) + (y3 - y2) \* (y3 - y2));

ac = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p1 = (ab + ac + bc) / 2;

double s1 = sqrt(p1 \* (p1 - ab) \* (p1 - bc) \* (p1 - ac));

//std::cout << s << " " << s1 << std::endl;

if (s <= s1)

{

return 1;

}

else

{

return 0;

}

}

int main()

{

system("color f1");

int x1, y1, x2, y2, x3, y3;

x1 = 1; y1 = 0;

x2 = 7; y2 = 0;

x3 = 1; y3 = 5;

Triangle first\_triangle(x1, y1, x2, y2, x3, y3);

std::cout<<"First triangle: {("<<x1<<", "<<y1<<"); "<< "("<<x2<<", "<<y2<<"); ("<<x3<<", "<<y3<<")} "<< std::endl ;

x1 = 2; y1 = 5;

x2 = 0; y2 = 11;

x3 = 3; y3 = 2;

Triangle second\_triangle(x1, y1, x2, y2, x3, y3);

std::cout << "Second triangle: {(" << x1 << ", " << y1 << "); " << "(" << x2 << ", " << y2 << "); (" << x3 << ", " << y3 << ")} " << std::endl;

int px, py;

px = 0; py = 1;

double point[2] = { px,py };

std::cout<<"can exist: " << (~first\_triangle) << std::endl;

std::cout<< "point ("<<px<<", "<< py << ") in triangle: " << first\_triangle + point << std::endl;

std::cout<<"S of first biger than second: " << (first\_triangle > second\_triangle) << std::endl;

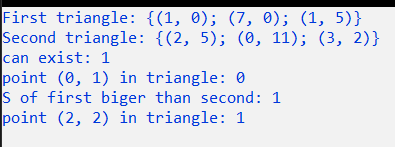
px = 2; py = 2;

double point2[2] = { px,py };

std::cout << "point (" << px << ", " << py << ") in triangle: " << first\_triangle + point2 << std::endl;

}

**Результат работы программы для входных параметров**



**Вывод**

Я освоил работу с перегруженными операторами и научился делать их самостоятельно