ФГАОУВО «Южно-Уральский государственный университет (НИУ)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине «Объектно-ориентированное программирование»

Автор работы

студент группы ЕТ-212

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.С.Никитин

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Работа зачтена с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.К.Демидов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Челябинск, 2022

# 1 Постановка задачи

I. Реализовать класс

15. Вектор в полярных координатах (r,phi в градусах) Polar

Конструктор: Polar(r,phi)

Операции:

x+y, x-y,

x+=d, x-=d (поворот вектора на d градусов),

x\*=d, x/=d (увеличение/уменьшение вектора в d раз),

d\*x (увеличенный в d раз вектор x, x не изменять!),

x==y, x!=y,

!x (x является нулевым вектором)

где x,y - вектора в полярных координатах, d - double

вывод, ввод в виде 5.1@30.5

Методы:

double getLength();

double getAngle();

Операции (если есть в задании) =, [], +=, -=, \*=, /=, префиксные ++, -- определять как методы.  
Для ввода переопределить >> , для вывода - << . Формат ввода-вывода объектов делать так, как указано в задании.  
Запись [текст] означает, что текст может отсутствовать, например, конструктор ИмяКласса(a[,b[,c]]) может быть вызван с 1, 2 или 3 аргументами.  
Пример:

Vector a(0,0),b(1.5,0.3);

cout<<"Введите вектор a:"<<endl;

cin>>a;

//нужно ввести (1.2,3.2) вместе со скобками и запятой

cout<<a<<" + "<<b<<" = "<<(a+b)<<endl;

// печатается (1.2,3.2) + (1.5,0.3) = (2.7,3.5)

II. Реализовать main c тестами (создание объектов и выполнение действий с ними)

# 2 Описание интерфейса класса

class Polar{

double r; // длина радиус-вектора

double phi; // угол

public:

Polar(double = 1, double = 0);

Polar &operator+=(double d);//перегрузка операции +=

Polar &operator-=(double d);//перегрузка операции -=

Polar &operator\*=(double d);//перегрузка операции \*=

Polar &operator/=(double d);//перегрузка операции /=

bool operator!();//перегрузка операции !

friend bool operator==(const Polar vec1, const Polar vec2) {

return (vec1.r == vec2.r && vec1.phi == vec2.phi);

}//перегрузка операции ==

friend istream &operator>>(istream &, Polar &); // перегрузка операции >>

friend ostream &operator<<(ostream &, const Polar &); //перегрузка операции <<

double getLength() const {return r;}; // получить радиус

double getAngle() const {return phi;}; // получить угол

double recur\_angle(double);

};};

# 3 Описание тестов для проверки классов

int main(){

Polar vector1(8.5, 390);

Polar vector2(10.321, 60.432);

cout << "Вывод\n";

cout << "vector1 = "<< vector1 << " vector2 = " << vector2<< "\n";

cout << "Ввод vector2\n";

cout << "vector2 = ";

cin >> vector2;

cout << "Проверка операции !vector2\n";

cout << " !" << vector2 << " = " << !vector2 << "\n";

cout << "Проверка операции vector1 == vector2\n";

cout << vector1 << " == " << vector2 << " = " << (vector1 == vector2) << "\n";

cout << "Проверка операции vector1 != vector2\n";

cout << vector1 << " != " << vector2 << " = " << (vector1 != vector2) << "\n";

cout << "Проверка операции vector2 += d\n";

cout << vector2 << " += " << 15 << " = ";

cout << (vector2 += 15) << "\n";

cout << "Проверка операции vector2 -= d\n";

cout << vector2 << " -= " << 20 << " = ";

cout << (vector2 -= 20) << "\n";

cout << "Проверка операции vector2 \*= d\n";

cout << vector2 << " \*= " << 3 << " = ";

cout << (vector2 \*= 3) << "\n";

cout << "Проверка операции vector2 /= d\n";

cout << vector2 << " /= " << 2 << " = ";

cout << (vector2 /= 2) << "\n";

cout << "Проверка операции d\* vector2\n";

cout << 2 << " \* " << vector2 << " = ";

Polar buf = 2 \* vector2;

cout << buf << "\n";

cout << "Проверка операции vector1+vector2\n";

cout << vector1 << " + " << vector2 << " = ";

Polar vector3 = vector1 + vector2;

cout << vector3 << "\n";

cout << "Проверка операции vector1-vector2\n";

cout << vector1 << " - " << vector2 << " = ";

vector3 = vector1 - vector2;

cout << vector3 << "\n";

return 0;

}**Полученные результаты.**

Вывод

vector1 = 8.5@30 vector2 = 10.321@60.432

Ввод vector2

vector2 = 2.4 54

Проверка операции !vector2

!2.4@4 = 0

Проверка операции vector1 == vector2

8.5@30 == 2.4@4 = 0

Проверка операции vector1 != vector2

8.5@30 != 2.4@4 = 1

Проверка операции vector2 += d

2.4@4 += 15 = 2.4@19

Проверка операции vector2 -= d

2.4@19 -= 20 = 2.4@-1

Проверка операции vector2 \*= d

2.4@-1 \*= 3 = 7.2@-1

Проверка операции vector2 /= d

7.2@-1 /= 2 = 3.6@-1

Проверка операции d\* vector2

2 \* 3.6@-1 = 7.2@-1

Проверка операции vector1+vector2

8.5@30 + 3.6@-1 = 15.5007@0.000645301

Проверка операции vector1-vector2

8.5@30 - 3.6@-1 = 5.31994@0.00150939

# 4 Листинг реализации класса

double Polar::recur\_angle(double angle){

double grad;

double modGrad = modf(angle,&grad);

return int(grad)%360+modGrad;

}

Polar::Polar(double r, double phi): r(r), phi(recur\_angle(phi)) {}

Polar &Polar:: operator +=(double angle){

phi = recur\_angle(phi + angle);

return \*this;

}

Polar &Polar:: operator -=(double angle){

phi = recur\_angle(phi - angle);

return \*this;

}

Polar &Polar:: operator \*=(double d){

r \*= d;

return \*this;

}

Polar &Polar:: operator /=(double d){

if(d != 0)

r /= d;

return \*this;

}

istream &operator>>(istream &i, Polar &vec){

char c;

return i >> vec.r >> c >> vec.phi;

}

ostream &operator<<(ostream &o, const Polar &vec){

return o << vec.r << '@' << vec.phi;

}

inline bool operator != (const Polar vec1, const Polar vec2){

return !(vec1 == vec2);

}

inline Polar operator \* (double d, Polar vec1){

return vec1 \*= d;

}

bool Polar:: operator!(){

return r == 0;

}

inline Polar operator + (Polar vec1, Polar vec2){

double angl1 = vec1.getAngle() \* TO\_RAD;

double angl2 = vec2.getAngle() \* TO\_RAD;

double x = vec1.getLength() \* cos(angl1) + vec2.getLength() \* cos(angl2);

double vector2 = vec1.getLength() \* sin(angl1) + vec2.getLength() \* sin(angl2);

double r = sqrt(x\*x + x\*x);

double phi = atan2(vector2, x) / TO\_RAD;

Polar res(r, phi);

return res;

}

inline Polar operator - (Polar vec1, Polar vec2) {

double angl1 = vec1.getAngle() \* TO\_RAD;

double angl2 = vec2.getAngle() \* TO\_RAD;

double x = vec1.getLength() \* cos(angl1) - vec2.getLength() \* cos(angl2);

double vector2 = vec1.getLength() \* sin(angl1) - vec2.getLength() \* sin(angl2);

double r = sqrt(x\*x + x\*x);

double phi = atan2(vector2, x) / TO\_RAD;

Polar res(r, phi);

return res;

}