#include "pch.h"

#include <iostream>

#include <math.h>

using namespace System;

using namespace std;

#define PI 3.141592

class unghi

{

public:

double u;

unghi(int v) :u(PI\* v / 180.) {}

operator double() { return u; }

};

class complex

{

public:

double Re, Im;

public:

complex(double r = 0., double i = 0.) : Re(r), Im(i) { }

friend ostream & operator<<( ostream &, complex);

complex operator+(complex z)

{

return complex(Re + z.Re, Im + z.Im);

}

complex operator-(complex z)

{

complex t; t.Re = Re - z.Re; t.Im = Im - z.Im; return t;

}

complex operator-()

{

return complex(-Re, -Im);

}

complex operator~() { return complex(Re, -Im); }

complex operator\*(complex z)

{

return complex(Re \* z.Re - Im \* z.Im, Re \* z.Im + Im \* z.Re);

}

double operator!() { return sqrt(Re \* Re + Im \* Im); }

operator double() { return !(\*this); }

complex operator/(complex z)

{

return complex((Re \* z.Re + Im \* z.Im) / (z.Re \* z.Re + z.Im \* z.Im),

(-Re \* z.Im + Im \* z.Re) / (z.Re \* z.Re + z.Im \* z.Im));

}

complex operator\*(double n)

{

return complex(Re \* n, Im \* n);

}

complex operator/(double n)

{

return complex(Re / n, Im / n);

}

complex operator+=(complex z)

{

return \*this = \*this + z;

}

friend bool operator==(complex z1, complex z2)

{

return (z1.Im - z2.Im < 0, 00001) &&

(z1.Re - z2.Re < 0, 00001) ? 1 : 0;

}

friend bool operator!=(complex z1, complex z2)

{

return !(z1 == z2);

}

};

ostream& operator<<(ostream& out, complex z)

{

out << z.Re << (z.Im < 0 ? "" : "+") << z.Im << "i"; return out;

}

int main() {

complex z, z1(1., -31.), z2(1., -1.), v, k;

double m; unghi u = 30;

z1 = complex(1, (unghi)30);

z2 = complex(0.5, sqrt(3.) / 2);

//cout << (z1==z2 ? "egale":"Diferite");

//cout <<(double)z1;

cout << (z += z1 += z2);

cout << endl;

}