NICOLETA RADU T3

Programare OOP / c++

# Numere Complexe

#include<iostream>

using namespace std;

class complex

{

private:

// real and imaginary parts, a(real) and b(imaginary)

double a = 0, b = 0;

public:

// constructori

complex(double);

complex();

complex(const complex\*);

complex(const complex&);

~complex()

{

cout << "Dealocare memorie" << endl;

}

complex sum(complex);

complex subtract(complex);

complex multiply(complex);

complex divide(complex);

private:

complex negativ();

complex conjugate();

double absolute();

};

inline complex::complex(double x): a(x),b(x)

{

a = x;

b = x;

cout << "Initializare prin parametri" << endl;

}

inline complex::complex()

{

cout << "Initializare fara parametri" << endl;

}

inline complex::complex(const complex\* x)

{

a = x->a;

b = x->b;

}

inline complex::complex(const complex& x)

{

a = x.a;

b = x.b;

}

complex complex::sum(complex A)

{

complex rezultat;

rezultat.a = a + A.a;

rezultat.b = b + A.b;

return rezultat;

}

complex complex::negativ()

{

complex rezultat{};

rezultat.a = -a;

rezultat.b = -b;

return rezultat;

}

complex complex::subtract(complex A)

{

complex rezultat{};

rezultat = sum(A.negativ());

return rezultat;

}

complex complex::multiply(complex A)

{

complex rezultat{};

rezultat.a = a \* A.a - b \* A.b;

rezultat.b = a \* A.b - b \* A.a;

return rezultat;

}

double complex::absolute()

{

return sqrt(a \* a + b \* b);

}

complex complex::conjugate()

{

complex rezultat;

rezultat.a = a;

rezultat.b = -b;

return rezultat;

}

complex complex::divide(complex A)

{

complex rezultat;

complex n;

n = multiply(A.conjugate());

rezultat.a = n.a / pow(A.absolute(), 2);

rezultat.b = n.b / pow(A.absolute(), 2);

return rezultat;

}

**APEL**

complex a(8), b(7);

complex rezultat;

a.write();

b.write();

rezultat = a.sum(b);

cout << "Suma: " << endl;

rezultat.write();

rezultat = a.divide(b);

cout << "Diferenta: " << endl;

rezultat.write();

rezultat = a.multiply(b);

cout << "Inmultire: " << endl;

rezultat.write();

rezultat = a.divide(b);

cout << "Impartire: " << endl;

rezultat.write();

# Polinom

#include <iostream>

#include <vector>

using namespace std;

class Polinom

{

private:

int grad;

vector <float> coeficient;

public:

Polinom(int);

~Polinom()

{

}

void citire();

void afisare();

void dealocare();

Polinom suma(Polinom);

float calculPunct(float);

};

inline Polinom::Polinom(int g)

{

grad = g;

//[x.grad + 1];

for (int i = 0; i < grad + 1; i++)

{

coeficient.push\_back(0);

}

}

void Polinom::dealocare()

{

grad = 0;

coeficient.clear();

}

void Polinom::citire()

{

cout << "Introduceti coeficientii polinomului" << endl;

for (int i = 0; i < grad; i++)

{

cout << "[" << i << "]= ";

cin >> coeficient.at(i);

}

}

void Polinom::afisare()

{

cout << coeficient.at(grad) << " x^" << grad;

for (int i = grad - 1; i >= 1 ; i--)

{

if (coeficient.at(i) > 0)

{

cout << " + " << coeficient.at(i) << " \* x^" << i << " ";

}

else

{

if (coeficient.at(i) < 0)

{

cout << coeficient.at(i) << " \* x^" << i << " ";

}

}

}

if (coeficient.at(0) > 0)

{

cout << " + " << coeficient.at(0) << " ";

}

else

{

if (coeficient.at(0) < 0)

{

cout << coeficient.at(0) << " ";

}

}

cout << endl;

}

Polinom Polinom::suma(Polinom x)

{

// cazul 1 - Gradele sunt egale

if (grad == x.grad)

{

Polinom rezultat(grad);

for (int i = 0; i < rezultat.grad; i++)

{

rezultat.coeficient.at(i) = coeficient.at(i) + x.coeficient.at(i);

}

return rezultat;

}

// cazul 2 - Gradul polinom. 1 este mai mare decat polinom 2

if (grad > x.grad)

{

Polinom rezultat(grad);

for (int i = 0; i < x.grad; i++)

{

rezultat.coeficient[i] = coeficient[i] + x.coeficient[i];

}

for (int i = x.grad + 1; i <= rezultat.grad; i++)

{

rezultat.coeficient[i] = coeficient[i];

}

return rezultat;

}

// cazul 3 - Gradul polinom. 2 mai mare decat polinom 1

if (x.grad > grad)

{

Polinom rezultat(x.grad);

for (int i = 0; i < grad; i++)

{

rezultat.coeficient[i] = coeficient[i] + x.coeficient[i];

}

for (int i = grad + 1; i <= rezultat.grad; i++)

{

rezultat.coeficient[i] = x.coeficient[i];

}

return rezultat;

}

}

float Polinom::calculPunct(float x)

{

float s = 0;

for (int i = 0; i < grad; i++)

{

s = s + coeficient[i] \* pow(x, (float)i);

}

return s;

}

**APEL**

int n;

cout << "Dati gradul polinoamelor: ";

cin >> n;

Polinom a(n);

Polinom b(n);

a.citire();

b.citire();

cout << endl;

a.afisare();

b.afisare();

Polinom rez(4);

rez = a.suma(b);

cout << "Suma dintre a si b" << endl;

rez.afisare();

cout << "Valoarea polinomului intr-un punct" << endl;

cout << "Dati punctul: ";

float x{};

cin >> x;

cout << "Valoarea lui a intr-un in " << x << " este " << a.calculPunct(x) << endl;

cout << endl;

