**Программный код**

#include <stdio.h>

#include <math.h>

using namespace std;

class Complex {

public:

Complex() {}

//конструктор по умолчанию

Complex(double re)

{

Complex::re = re;

Complex::im = 0;

}

//конструктор по инициализации

Complex(double re, double im)

{

Complex::re = re;

Complex::im = im;

}

~Complex() {}

//объявление константных функций

double Re(void) const { return re; }

double Im(void) const { return im; }

double R(void) const { return sqrt(re\*re + im \* im); }

double Phi(void) const { return atan2(re, im); }

friend ostream& operator<<(ostream &out, const Complex &rval) {

out << rval.Re() << " + (" << rval.Im() << ")i";

return out;

}

friend istream& operator>>(istream &in, Complex &rval) {

in >> rval.re >> rval.im;

return in;

}

Complex operator+(const Complex &rval) const {

return Complex(re + rval.Re(), im + rval.Im());

}

Complex operator-(const Complex &rval) const {

return Complex(re - rval.Re(), im - rval.Im());

}

Complex operator\*(const Complex &rval) const {

double i, j;

i = re \* rval.Re() - im \* rval.Im();

j = re \* rval.Im() + rval.Re() \* im;

Complex result(i, j);

return result;

}

Complex operator/(const Complex &rval) const {

double i, j, rad;

rad = re \* re + rval.Im() \* rval.Im();

i = (re \* rval.Re() + im \* rval.Im()) / rad;

j = (rval.Re() \* im - re \* rval.Im()) / rad;

Complex result(i, j);

return result;

}

//разработка пользовательских функций

Complex c\_exp(const Complex &z) const

{

double i, j;

i = exp(z.Re()) \* cos(z.Im());

j = exp(z.Re()) \* sin(z.Im());

Complex result(i, j);

return result;

}

Complex c\_sin(const Complex &z) const

{

double i, j;

i = sin(z.Re()) \* cosh(z.Im());

j = cos(z.Re()) \* sinh(z.Im());

Complex result(i, j);

return result;

}

Complex c\_cos(const Complex &z) const

{

double i, j;

i = cos(z.Re()) \* cosh(z.Im());

j = -(sin(z.Re()) \* sinh(z.Im()));

Complex result(i, j);

return result;

}

Complex c\_ch(const Complex &z) const

{

Complex f1(1.0);

Complex f2(2.0);

return ((c\_exp(z) + (f1 / c\_exp(z))) / f2);

}

Complex c\_sh(const Complex &z) const

{

Complex f1(1.0);

Complex f2(2.0);

return ((c\_exp(z) - (f1 / c\_exp(z))) / f2);

}

Complex c\_pow(const Complex &z1, const Complex &z2) const

{

double i, j, n;

n = z2.Re();

i = pow(z1.R(), n) \* cos(n\*z1.Phi());

j = pow(z1.R(), n) \* sin(n\*z1.Phi());

Complex result(i, j);

return result;

}

private:

double re;

double im;

};

Complex y(const Complex &z)

{

Complex f(2.0);

return (f.c\_pow(z, f) - f.c\_cos(z\*f));

}

int main(int argc, const char \*\* argv)

{

Complex z;

in >> z;

out << "Пользовательская функция = " << y(z) << end;

out << "Sin(z) = " << z.c\_sin(z) << end;

out << "Cos(z) = " << z.c\_cos(z) << end;

out << "ch(z) = " << z.c\_ch(z) << end;

out << "sh(z) = " << z.c\_sh(z) << end;

out << "Exp(z) = " << z.c\_exp(z) << end;

z.~Complex();

return 0;

}