

Mar 19, 10 16:25

exer56v0.cpp

Page 1/2

```

/*
    Exercício 56, Folha 7

    Implemente de uma classe apropriada para representar os números
    complexos

    versão 0 - tudo num só ficheiro.
*/

#include <iostream>
using namespace std;

class Complexos {
public:
    // Construtores
    Complexos (double x, double y){
        re=x;
        img=y;
    }

    Complexos (double x){};
    Complexos (){};

    Complexos realPuro(double x) {
        Complexos res;
        res.re=x;
        res.img=0;
        return(res);
    }

    Complexos imagPuro(double y) {
        Complexos res;
        res.re=0;
        res.img=y;
        return(res);
    }

    // Manipuladores
    double parteReal() {
        return(re);
    }

    double parteImag() {
        return(img);
    }

    Complexos soma(Complexos a){
        Complexos res;
        res.re = re+a.re;
        res.img = img+a.img;
        return(res);
    }

    Complexos produto(Complexos a){
        Complexos res;
        res.re = re+a.re;
        res.img = img+a.img;
        return(res);
    }

    // inverso
    Complexos inverso() {
        Complexos res;
        double div;
        div = re*re+img*img;
        res.re = re/div;
        res.img = -img/div;
    }

```

Mar 19, 10 16:25

exer56v0.cpp

Page 2/2

```

        return(res);
    }

    // simetrico
    Complexos simetrico() {
        Complexos res;
        res.re = -re;
        res.img = -img;
        return(res);
    }

    // conjugado
    Complexos conjugado() {
        Complexos res;
        res.re = re;
        res.img = -img;
        return(res);
    }

    /*
        Relações
    */
    // igualdade (sem sobrecarga de operadores)
    bool igualdade(Complexos a) {
        return(a.re==re && a.img==img);
    }
    // desigualdade (sem sobrecarga de operadores)
    bool desigualdade(Complexos a) {
        return(a.re!=re || a.img!=img);
    }

private:
    // Complexo como dois reais, parte real e parte imaginária
    double re,img;
};

int main() {
    double x,y;

    // leitura
    cout << "Introduza um complexo (dois reais): ";

    cin >> x >> y;

    // Inicialização do objecto
    Complexos z(x,y);

    // escrita do resultado
    cout << "\nz=" << z.parteReal() << "+i" << z.parteImag() << endl;
}

```

Mar 19, 10 19:44

exer56v1.cpp

Page 1/2

```

/*
    Exercício 56, Folha 7

    Implemente de uma classe apropriada para representar os números
    complexos

    versão 1 - com separação de ficheiro (usaComplexosV1.cpp,
    exer56v1.cpp, exer56v1.h)

*/

#include <iostream>
#include "exer56v1.h"

using namespace std;

// Construtores
Complexos::Complexos (double x, double y){
    re=x;
    img=y;
}

Complexos::Complexos (double x){};
Complexos::Complexos (){};

Complexos::Complexos Complexos::realPuro(double x) {
    Complexos res;
    res.re = x;
    res.img = 0;
    return(res);
}

Complexos::Complexos Complexos::imagPuro(double y) {
    Complexos res;
    res.re = 0;
    res.img = y;
    return(res);
}

// Manipuladores
double Complexos::parteReal() {
    return(re);
}

double Complexos::parteImag() {
    return(img);
}

Complexos::Complexos Complexos::soma(Complexos a){
    Complexos res;
    res.re = re+a.re;
    res.img = img+a.img;
    return(res);
}

Complexos::Complexos Complexos::produto(Complexos a){
    Complexos res;
    res.re = re+a.re;
    res.img = img+a.img;
    return(res);
}

// inverso
Complexos::Complexos Complexos::inverso() {
    Complexos res;
    double div;
    div = re*re+img*img;

```

Mar 19, 10 19:44

exer56v1.cpp

Page 2/2

```

    res.re = re/div;
    res.img = -img/div;
    return(res);
}

// simetrico
Complexos::Complexos Complexos::simetrico() {
    Complexos res;
    res.re = -re;
    res.img = -img;
    return(res);
}

// conjugado
Complexos::Complexos Complexos::conjugado() {
    Complexos res;
    res.re = re;
    res.img = -img;
    return(res);
}

/*
    Relações
*/

// igualdade (sem sobrecarga de operadores)
bool Complexos::igualdade(Complexos a) {
    return(a.re==re && a.img==img);
}

// desigualdade (sem sobrecarga de operadores)
bool Complexos::desigualdade(Complexos a) {
    return(a.re!=re || a.img!=img);
}

```

Mar 19, 10 16:25

exer56v1.h

Page 1/1

```

class Complexos {
public:
    // Construtores
    Complexos(double, double);
    // invalida os construtores com menos de duas componentes
    Complexos(double);
    Complexos();
    // falsos construtores
    Complexos realPuro(double);
    Complexos imagPuro(double);
    /*
        Manipuladores
    */
    /*
        Projecções
    */
    double parteReal();
    double parteImag();
    /*
        operações elementares
    */
    // soma
    Complexos soma(Complexos);
    // produto
    Complexos produto(Complexos);
    // inverso
    Complexos inverso();
    // simetrico
    Complexos simetrico();
    // conjugado
    Complexos conjugado();
    /*
        Relações
    */
    // igualdade (sem sobrecarga de operadores)
    bool igualdade(Complexos);
    // desigualdade (sem sobrecarga de operadores)
    bool desigualdade(Complexos);
protected:
    // Complexo como dois reais, parte real e parte imaginária
    double re, img;
};

```

Mar 19, 10 16:28

usaComplexosV1.cpp

Page 1/1

```

#include <iostream>
#include "exer56v1.h"

using namespace std;

int main() {
    double x,y;
    Complexos z; //declaração

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z1(x,y);

    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z2(x,y);

    // soma dos complexos
    z = z1.soma(z2);

    // escrita do resultado
    cout << "\nz=" << z.parteReal() << "+i" << z.parteImag() << endl;
}

/*
exer56v1: exer56v1.o usaComplexos.o
g++ -o usaComplexos exer56v1.o usaComplexos.cpp

*/

```

Mar 19, 10 16:31

exer56v2.cpp

Page 1/2

```

/*
    Exercício 56, Folha 7

    Implemente de uma classe apropriada para representar os números
    complexos

    versão 2
    - com separação de ficheiro (usaComplexosV2.cpp, exer56v2.cpp,
      exer56v2.h)
    - com sobrecarga dos identificadores referentes aos operadores.
*/

#include <iostream>
#include "exer56v2.h"

using namespace std;

// Construtores
Complexos::Complexos (double x, double y){
    re=x;
    img=y;
}

Complexos::Complexos (double x){};
Complexos::Complexos (){};

Complexos::Complexos Complexos::realPuro(double x) {
    Complexos res;
    res.re = x;
    res.img = 0;
    return(res);
}

Complexos::Complexos Complexos::imagPuro(double y) {
    Complexos res;
    res.re = 0;
    res.img = y;
    return(res);
}

// Manipuladores
double Complexos::parteReal() {
    return(re);
}

double Complexos::parteImag() {
    return(img);
}

Complexos::Complexos operator+ (Complexos a, Complexos b){
    return(Complexos(a.re+b.re,a.img+b.img));
}

Complexos::Complexos operator- (Complexos a, Complexos b){
    return(Complexos(a.re-b.re,a.img-b.img));
}

Complexos::Complexos operator- (Complexos a){
    return(Complexos(-a.re,-a.img));
}

Complexos::Complexos operator* (Complexos a, Complexos b){
    return(Complexos(a.re*b.re-a.img*b.img,a.re*b.img+b.re*a.img));
}

// inverso
Complexos::Complexos Complexos::inverso() {
    Complexos res;

```

Mar 19, 10 16:31

exer56v2.cpp

Page 2/2

```

    double div;
    div = re*re+img*img;
    res.re = re/div;
    res.img = -img/div;
    return(res);
}

Complexos::Complexos operator/(Complexos a, Complexos b){
    Complexos res;
    double div;
    div = b.re*b.re+b.img*b.img;
    res.re = b.re/div;
    res.img = -b.img/div;
    res.re = res.img;
    return(Complexos(a.re*res.re-a.img*res.img,a.re*res.img+res.re*a.img));
}

// conjugado
Complexos::Complexos Complexos::conjugado() {
    Complexos res;
    res.re = re;
    res.img = -img;
    return(res);
}

/*
    Relações
*/
// igualdade
bool operator== (Complexos a,Complexos b) {
    return(a.re==b.re && a.img==b.img);
}

// desigualdade
bool operator!= (Complexos a,Complexos b) {
    return(a.re!=b.re || a.img!=b.img);
}

```

Mar 19, 10 16:08

exer56v2.h

Page 1/1

```

class Complexos {
public:
    // Construtores
    Complexos(double, double);
    // invalida os construtores com menos de duas componentes
    Complexos(double);
    Complexos();
    // falsos construtores
    Complexos realPuro(double);
    Complexos imagPuro(double);
    /*
        Manipuladores
    */
    /*
        Projecções
    */
    double parteReal();
    double parteImag();
    /*
        operações elementares (com sobrecarga de operadores)
    */
    // soma
    friend Complexos operator+ (Complexos,Complexos);
    // diferença
    friend Complexos operator- (Complexos,Complexos); // binário
    // simétrico
    friend Complexos operator- (Complexos); // unário
    // produto
    friend Complexos operator* (Complexos,Complexos);
    // divisão
    friend Complexos operator/ (Complexos,Complexos);
    // inverso
    Complexos inverso();
    // conjugado
    Complexos conjugado();
    /*
        Relações
    */
    // igualdade (com sobrecarga de operadores)
    friend bool operator==(Complexos,Complexos);
    // desigualdade (com sobrecarga de operadores)
    friend bool operator!=(Complexos,Complexos);
protected:
    // Complexo como dois reais, parte real e parte imaginária
    double re,img;
};

```

Mar 19, 10 15:19

usaComplexosV2.cpp

Page 1/1

```

#include <iostream>
#include "exer56v2.h"

using namespace std;

int main() {
    double x,y;

    Complexos z; //declaração

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z1(x,y);

    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z2(x,y);

    // soma dos complexos
    z = z1+z2;

    // escrita do resultado
    cout << "\nz=" << z.parteReal() << "+i" << z.parteImag() << endl;
}

```

Mar 19, 10 16:32

exer56v3.cpp

Page 1/2

```

/*
    Exercício 56, Folha 7

    Implemente de uma classe apropriada para representar os números
    complexos

    versão 3
    - com separação de ficheiro (usaComplexosV3.cpp, exer56v3.cpp,
      exer56v3.h)
    - com sobrecarga dos identificadores referentes aos operadores.
    - utilizando namespaces

*/

#include <iostream>
#include "exer56v3.h"

using namespace std;

namespace exercP00{
// Construtores
Complexos::Complexos (double x, double y){
    re=x;
    img=y;
}

Complexos::Complexos (double x){};
Complexos::Complexos (){};

Complexos::Complexos Complexos::realPuro(double x) {
    Complexos res;
    res.re = x;
    res.img = 0;
    return(res);
}

Complexos::Complexos Complexos::imagPuro(double y) {
    Complexos res;
    res.re = 0;
    res.img = y;
    return(res);
}

// Manipuladores
double Complexos::parteReal() {
    return(re);
}

double Complexos::parteImag() {
    return(img);
}

Complexos::Complexos operator+ (Complexos a, Complexos b){
    return(Complexos(a.re+b.re,a.img+b.img));
}

Complexos::Complexos operator- (Complexos a, Complexos b){
    return(Complexos(a.re-b.re,a.img-b.img));
}

Complexos::Complexos operator- (Complexos a){
    return(Complexos(-a.re,-a.img));
}

Complexos::Complexos operator* (Complexos a, Complexos b){
    return(Complexos(a.re*b.re-a.img*b.img,a.re*b.img+b.re*a.img));
}

// inverso

```

Mar 19, 10 16:32

exer56v3.cpp

Page 2/2

```

Complexos::Complexos Complexos::inverso() {
    Complexos res;
    double div;
    div = re*re+img*img;
    res.re = re/div;
    res.img = -img/div;
    return(res);
}

Complexos::Complexos operator/(Complexos a, Complexos b){
    Complexos res;
    double div;
    div = b.re*b.re+b.img*b.img;
    res.re = b.re/div;
    res.img = -b.img/div;
    res.re = res.img;
    return(Complexos(a.re*res.re-a.img*res.img,a.re*res.img+res.re*a.img));
}

// conjugado
Complexos::Complexos Complexos::conjugado() {
    Complexos res;
    res.re = re;
    res.img = -img;
    return(res);
}

/*
    Relações
*/
// igualdade
bool operator== (Complexos a,Complexos b) {
    return(a.re==b.re && a.img==b.img);
}

// desigualdade
bool operator!= (Complexos a,Complexos b) {
    return(a.re!=b.re || a.img!=b.img);
}

```

Mar 19, 10 15:25

exer56v3.h

Page 1/1

```

namespace exercP00{
class Complexos {
public:
    // Construtores
    Complexos (double, double);
    // invalida os construtores com menos de duas componentes
    Complexos (double);
    Complexos ();
    // falsos construtores
    Complexos realPuro(double);
    Complexos imagPuro(double);
    /*
        Manipuladores
    */
    /*
        Projecções
    */
    double parteReal();
    double parteImag();
    /*
        operações elementares
    */
    // soma
    friend Complexos operator+ (Complexos,Complexos);
    // diferença
    friend Complexos operator- (Complexos,Complexos); // binário
    // simétrico
    friend Complexos operator- (Complexos); // unário
    // produto
    friend Complexos operator* (Complexos,Complexos);
    // divisao
    friend Complexos operator/ (Complexos,Complexos);
    // inverso
    Complexos inverso();
    // conjugado
    Complexos conjugado();
    /*
        Relações
    */
    // igualdade (sem sobrecarga de operadores)
    friend bool operator==(Complexos,Complexos);
    friend bool operator!=(Complexos,Complexos);
protected:
    // Complexo como dois reais, parte real e parte imaginária
    double re,img;
};
}

```

Mar 19, 10 15:41

usaComplexosV3.cpp

Page 1/1

```

#include <iostream>
#include "exer56v3.h"

// espaço de nomes "standard"
using namespace std;
// espaço de nomes para P00
using namespace exercP00;

int main() {
    double x,y;

    Complexos z; //declaração

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z1(x,y);

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z2(x,y);

    // soma dos complexos
    z = z1+z2;

    // escrita do resultado
    cout << "\nz=" << z.parteReal() << "+i" << z.parteImag() << endl;
}

```

Mar 19, 10 16:32

exer56v4.cpp

Page 1/2

```

/*
    Exercício 56, Folha 7

    Implemente de uma classe apropriada para representar os números
    complexos

    versão 4
    - com separação de ficheiro (usaComplexosV4.cpp, exer56v4.cpp,
      exer56v4.h)
    - com sobrecarga dos identificadores referentes aos operadores.
    - utilizando namespaces

    - Uma implementação diferente, complexos como um vector com duas
      componentes. O programa de chamada não se altera.
*/

#include <iostream>
#include "exer56v4.h"

using namespace std;

namespace exercPOO{
    // Construtores
    Complexos::Complexos (double x, double y){
        cmplx[0]=x;
        cmplx[1]=y;
    }

    Complexos::Complexos (double x){};
    Complexos::Complexos (){};

    // Manipuladores
    double Complexos::parteReal() {
        return(cmplx[0]);
    }

    double Complexos::parteImag() {
        return(cmplx[1]);
    }

    Complexos::Complexos operator+ (Complexos a, Complexos b){
        return(Complexos(a.cmplx[0]+b.cmplx[0],a.cmplx[1]+b.cmplx[1]));
    }

    Complexos::Complexos operator- (Complexos a, Complexos b){
        return(Complexos(a.cmplx[0]-b.cmplx[0],a.cmplx[1]-b.cmplx[1]));
    }

    Complexos::Complexos operator- (Complexos a){
        return(Complexos(-a.cmplx[0],-a.cmplx[1]));
    }

    Complexos::Complexos operator* (Complexos a, Complexos b){
        return(Complexos(a.cmplx[0]*b.cmplx[0]-a.cmplx[1]*b.cmplx[1],a.cmplx[0]*b.cmplx[1]+b.cmplx[0]*a.cmplx[1]));
    }

    // inverso
    Complexos::Complexos Complexos::inverso() {
        Complexos res;
        double div;
        div = cmplx[0]*cmplx[0]+cmplx[1]*cmplx[1];
        res.cmplx[0] = cmplx[0]/div;
        res.cmplx[1] = -cmplx[1]/div;
        return(res);
    }

    Complexos::Complexos operator/(Complexos a, Complexos b){

```

Mar 19, 10 16:32

exer56v4.cpp

Page 2/2

```

        Complexos res;
        double div;
        div = b.cmplx[0]*b.cmplx[0]+b.cmplx[1]*b.cmplx[1];
        res.cmplx[0] = b.cmplx[0]/div;
        res.cmplx[1] = -b.cmplx[1]/div;
        res.cmplx[0] = res.cmplx[1];
        return(Complexos(a.cmplx[0]*res.cmplx[0]-a.cmplx[1]*res.cmplx[1],a.cmplx[0]*
        res.cmplx[1]+res.cmplx[0]*a.cmplx[1]));
    }

    // conjugado
    Complexos::Complexos Complexos::conjugado() {
        Complexos res;
        res.cmplx[0] = cmplx[0];
        res.cmplx[1] = -cmplx[1];
        return(res);
    }

    /*
        Relações
    */
    // igualdade
    bool operator== (Complexos a,Complexos b) {
        return(a.cmplx[0]==b.cmplx[0] && a.cmplx[1]==b.cmplx[1]);
    }

    // desigualdade
    bool operator!= (Complexos a,Complexos b) {
        return(a.cmplx[0]!=b.cmplx[1] || a.cmplx[1]!=b.cmplx[1]);
    }
}

```


Mar 19, 10 16:12

exer56v4.h

Page 1/1

```

namespace exercP00{
class Complexos {
public:
    // Construtores
    Complexos (double, double);
    // invalida os construtores com menos de duas componentes
    Complexos (double);
    Complexos ();
    /*
        Manipuladores
    */
    /*
        Projecções
    */
    double parteReal();
    double parteImag();
    /*
        operações elementares
    */
    // soma
    friend Complexos operator+ (Complexos,Complexos);
    // diferença
    friend Complexos operator- (Complexos,Complexos); // binário
    // simétrico
    friend Complexos operator- (Complexos); // unário
    // produto
    friend Complexos operator* (Complexos,Complexos);
    // divisao
    friend Complexos operator/ (Complexos,Complexos);
    // inverso
    Complexos inverso();
    // conjugado
    Complexos conjugado();
    /*
        Relações
    */
    // igualdade (sem sobrecarga de operadores)
    friend bool operator==(Complexos,Complexos);
    friend bool operator!=(Complexos,Complexos);
protected:
    // Complexo como um vector de duas componentes
    double cmplx[2];
};
}

```

Mar 19, 10 19:42

usaComplexosV4.cpp

Page 1/1

```

#include <iostream>
#include "exer56v4.h"

// espaço de nomes "standard"
using namespace std;
// espaço de nomes para P00
using namespace exercP00;

int main() {
    double x,y;

    Complexos z; //declaração

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z1(x,y);

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z2(x,y);

    // soma dos complexos
    z = z1+z2;

    // escrita do resultado
    cout << "\nz=" << z.parteReal() << "+i" << z.parteImag() << endl;
}

```

Mar 19, 10 19:55

exer56v5.cpp

Page 1/2

```

/*
    Exercício 56, Folha 7

    Implemente de uma classe apropriada para representar os números
    complexos

    versão 5
    - com separação de ficheiro (usaComplexosV5.cpp, exer56v5.cpp,
      exer56v5.h)
    - com sobrecarga dos identificadores referentes aos operadores.
    - utilizando namespaces

    - Uma implementação diferente, complexos como um ponteiro para
      reais (vector com dimensão dinâmica). O programa de
      chamada não se altera.
*/

#include <iostream>
#include "exer56v5.h"

using namespace std;

namespace exercP00{
    // Construtores
    Complexos::Complexos (double x, double y){
        cmplx = new double [2];
        cmplx[0]=x;
        cmplx[1]=y;
    }

    Complexos::Complexos (double x){
        cmplx = new double [2];
        cmplx[0]=x;
        cmplx[1]=0;
    }

    Complexos::Complexos (){
        cmplx = new double [2];
        cmplx[0]=0;
        cmplx[1]=0;
    }

    // Manipuladores
    double Complexos::parteReal() {
        return(cmplx[0]);
    }

    double Complexos::parteImag() {
        return(cmplx[1]);
    }

    Complexos::Complexos operator+ (Complexos a, Complexos b){
        return(Complexos(a.cmplx[0]+b.cmplx[0],a.cmplx[1]+b.cmplx[1]));
    }

    Complexos::Complexos operator- (Complexos a, Complexos b){
        return(Complexos(a.cmplx[0]-b.cmplx[0],a.cmplx[1]-b.cmplx[1]));
    }

    Complexos::Complexos operator- (Complexos a){
        return(Complexos(-a.cmplx[0],-a.cmplx[1]));
    }

    Complexos::Complexos operator* (Complexos a, Complexos b){
        return(Complexos(a.cmplx[0]*b.cmplx[0]-a.cmplx[1]*b.cmplx[1],a.cmplx[0]*b.cmplx[1]+b.cmplx[0]*a.cmplx[1]));
    }
}

```

Mar 19, 10 19:55

exer56v5.cpp

Page 2/2

```

// inverso
Complexos::Complexos Complexos::inverso() {
    Complexos res;
    double div;
    div = cmplx[0]*cmplx[0]+cmplx[1]*cmplx[1];
    res.cmplx[0] = cmplx[0]/div;
    res.cmplx[1] = -cmplx[1]/div;
    return(res);
}

Complexos::Complexos operator/(Complexos a, Complexos b){
    Complexos res;
    double div;
    div = b.cmplx[0]*b.cmplx[0]+b.cmplx[1]*b.cmplx[1];
    res.cmplx[0] = a.cmplx[0]/div;
    res.cmplx[1] = -a.cmplx[1]/div;
    res.cmplx[0] = res.cmplx[0];
    return(Complexos(a.cmplx[0]*res.cmplx[0]-a.cmplx[1]*res.cmplx[1],a.cmplx[0]*
res.cmplx[1]+res.cmplx[0]*a.cmplx[1]));
}

// conjugado
Complexos::Complexos Complexos::conjugado() {
    Complexos res;
    res.cmplx[0] = cmplx[0];
    res.cmplx[1] = -cmplx[1];
    return(res);
}

/*
    Relações
*/
// igualdade
bool operator== (Complexos a,Complexos b) {
    return(a.cmplx[0]==b.cmplx[0] && a.cmplx[1]==b.cmplx[1]);
}

// desigualdade
bool operator!= (Complexos a,Complexos b) {
    return(a.cmplx[0]!=b.cmplx[1] || a.cmplx[1]!=b.cmplx[1]);
}
}

```

Mar 19, 10 19:54

exer56v5.h

Page 1/1

```

namespace exercP00{
class Complexos {
public:
    // Construtores
    Complexos (double, double);
    // invalida os construtores com menos de duas componentes
    Complexos (double);
    Complexos ();
    /*
        Manipuladores
    */
    /*
        Projecções
    */
    double parteReal();
    double parteImag();
    /*
        operações elementares
    */
    // soma
    friend Complexos operator+ (Complexos,Complexos);
    // diferença
    friend Complexos operator- (Complexos,Complexos); // binário
    // simétrico
    friend Complexos operator- (Complexos); // unário
    // produto
    friend Complexos operator* (Complexos,Complexos);
    // divisao
    friend Complexos operator/ (Complexos,Complexos);
    // inverso
    Complexos inverso();
    // conjugado
    Complexos conjugado();
    /*
        Relações
    */
    // igualdade (sem sobrecarga de operadores)
    friend bool operator==(Complexos,Complexos);
    friend bool operator!=(Complexos,Complexos);
protected:
    // Complexo como um vector de duas componentes
    double * cmplx;
};
}

```

Mar 19, 10 19:42

usaComplexosV5.cpp

Page 1/1

```

#include <iostream>
#include "exer56v5.h"

// espaço de nomes "standard"
using namespace std;
// espaço de nomes para P00
using namespace exercP00;

int main() {
    double x,y;

    Complexos z; //declaração

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z1(x,y);

    // leitura
    cout << "\nIntroduza um complexo (dois reais): ";
    cin >> x >> y;

    // Inicialização do objecto
    Complexos z2(x,y);

    // soma dos complexos
    z = z1+z2;

    // escrita do resultado
    cout << "\nz=" << z.parteReal() << "+i" << z.parteImag() << endl;
}

```