ECOP13A – Lab4

ECOP13A - Programação Orientada a Objetos
Prof. André Bernardi
andrebernardi@unifei.edu.br

Universidade Federal de Itajubá

4º Laboratório ECOP13A
30 abril 2025

1ª Questão

1ª: Criar uma classe que represente um tipo abstrato de dados **Complexo** com as seguintes características:

a. Possua duas variáveis do tipo double para representar a parte **real** e a parte **imaginária**.
b. Possua métodos que permitam que objetos desse tipo sejam *somados*, *subtraídos*, *multiplicados* e *divididos*.
c. Criar construtores que permitam a inicialização de objetos com e sem parâmetros.
d. Construir uma variável que pode ser utilizada para ser um contador do número de complexos que estão instanciadas em determinado momento em um programa.
e. Crie uma função para calcular e retornar o modulo do número complexo.
f. Criar uma função para imprimir um número complexo no formato “ a + b i ”

Criar um programa que teste as funcionalidades implementadas nos itens acima.

1ª questão – Exemplo de Solução

```
// Exercício 1 - arquivo complexo.h

#ifndef COMPLEXO_H
#define COMPLEXO_H

class Complexo
{
    private:
        double real, imag;
        static int n;

    public:
        Complexo() {
            real = 1;
            imag = 1;
            n++;
        }

        Complexo(double a, double b) {
            real = a;
            imag = b;
            n++;
        }

        Complexo(const Complexo& c) {
            real = c.real;
            imag = c.imag;
            n++;
        }

        ~Complexo() {n--;}

        Complexo somar(Complexo);
        Complexo subtrair(Complexo);
        Complexo multiplicar(Complexo);
        Complexo dividir(Complexo);

        void setReal(double a) {real = a;}
        void setImaginario(double a) {imag = a;}
        double getReal() {return real;}
        double getImaginario() {return imag;}
        int getObjetos() {return n;}

        double modulo();
        void print();

};

#endif
```

```cpp
#include <iostream>
#include <cmath>
#include "complexo.h"

using namespace std;

// inicialização do membro estático
int Complexo::n = 0;

//função somar - recebe um complexo como parâmetro e retorna um complexo
Complexo Complexo::somar(Complexo _complexo)
{
    Complexo temp(getReal() + _complexo.getReal(),
                  getImaginario() + _complexo.getImaginario());
    return temp;
}
//função subtrair - recebe um complexo como parâmetro e retorna um complexo
Complexo Complexo::subtrair(Complexo _complexo)
{
    Complexo temp(getReal() - _complexo.getReal(),
                    getImaginario() - _complexo.getImaginario());
    return temp;
}
//função multiplicar - recebe um complexo como parâmetro e retorna um complexo
Complexo Complexo::multiplicar(Complexo _complexo)
{
    Complexo temp(
        getReal()*_complexo.getReal() - getImaginario()*_complexo.getImaginario(),
        getReal()*_complexo.getImaginario() + getImaginario()*_complexo.getReal());
    return temp;
}
```


```
//função dividir - recebe um complexo como parametro e retorna um complexo
Complexo Complexo::dividir(Complexo _c)
{
   double a = ( real *_c.real + imag *_c.imag ) /
              (pow(_c.real, 2) + pow(_c.imag, 2));
   double b = (_c.real * imag - real *_c.imag ) /
              (pow(_c.real, 2) + pow(_c.imag, 2));
   Complexo temp(a, b);
   return temp;
}

//função modulo - calcula o modulo do complexo
double Complexo::modulo()
{
   return sqrt(pow(getReal(), 2) + pow(getImaginario(), 2));
}

//função print - imprime um complexo com o formato desejado
void Complexo::print()
{
   cout << getReal() << " ";
   if(getImaginario() < 0) cout << getImaginario() << "i" << endl;
   else cout << "+" << getImaginario() << "i" << endl;
}
```

Main

```
#include <iostream>
#include <cmath>
#include "complexo.h"

using namespace std;

int main()
{
    Complexo a(1,2);
    Complexo b(3,4);
    Complexo aux;
    int ans;

    cout << "A: ";
    a.print();
    cout << "B: ";
    b.print();

    cout << "A + B: ";
    aux = a.somar(b);
    aux.print();

    cout << "A - B: ";
    aux = a.subtrair(b);
    aux.print();

    cout << "A * B: ";
    aux = a.multiplicar(b);
    aux.print();

    cout << "A / B: ";
    aux = a.dividir(b);
    aux.print();

    cout << "|A|: " << a.modulo() << endl;
    cout << "|B|: " << b.modulo() << endl;

    cout << "Qt de objetos: " <<
              a.getObjetos() << endl;
}
```

2ª Questão

2ª Questão Escreva uma classe que represente um número inteiro longo com 30 dígitos. Acrescente funções que permitam que estes números possam ser lidos pelo teclado, impressos na tela, somados e subtraídos.

Declarar a classe completa com todos seus membros incluindo construtores e destrutor. Implementar a classe em um arquivo separado. **Sugestão**: utilize um vetor para armazenar cada um dos dígitos do número.

2ª questão – Exemplo de Solução

Ex. número 12.345.678.912.345 seria representado no vetor como:

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	1	2	3	4	5

Deste modo ele seria organizado no vetor como:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
5	4	3	2	1	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sendo os valores mais significativos, representados nos índices maiores.

Alternativas de Solução

- O vetor pode ser de números inteiros, mas para economizar memória ele pode ser também de caracteres.
- Caso seja de caracteres devemos lembrar os dígitos de ‘0’ a ‘9’ são armazenados na tabela ASCII a partir da posição 48, portanto devemos descontar o caractere ‘0’ para obter seus valores em inteiros.

Alternativas de Solução

- Caso seja de inteiros, devemos lembrar que ao fazer a leitura do número em forma de texto, também temos que descontar o valor do caractere ‘0’ para obter o valor numérico do digito.
- Para implementar o método de impressão desses números, é importante observar a forma como ele foi organizado dentro do vetor.

Alternativas de Solução

- Os métodos de soma e subtração, podem ser implementados seguindo o processo manual utilizado para realizar uma soma/subtração de vários dígitos, como por exemplo:

SOMA

																								1						1
29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	1	2	3	4	5
+																														
29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	1	2	3	4	5
29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	9	8	2	4	6	9	0

Ao calcular a soma, devemos levar em conta o “vai um”. soma = vet[i] + n.vet[i] + vaiUm;
Armazena-se o soma%10, em cada casa decimal do número.
Atualiza-se o **vaiUm** com soma/10.

2ª questão
Exemplo de Solução

```
#ifndef BIGINT_H
#define BIGINT_H

class BigInt
{
   private:
      int num[31];
      int len;

      void inicializar()
      {
         for(int i = 0; i < 31; i++) num[i] = 0;
      }

   public:
      BigInt() {inicializar();}
      ~BigInt() {};

      void leia();
      void print();
      BigInt soma(BigInt);
      BigInt subtrai(BigInt);
};
#endif
```

2ª questão
Exemplo de Solução

```
#include <iostream>
#include <string>
#include "bigint.h"

using namespace std;

void BigInt::leia()
{
   string a;
   cin >> a;
   len = a.length();
   for(int i = 0; i < len; i++)
   {
      num[i] = a[len-i-1] - '0';
   }
}

void BigInt::print()
{
   for(int i = len-1; i >= 0; i--)
      cout << num[i];
   cout << endl;
}
```

SOMA

```
BigInt BigInt::soma(BigInt b)
{
   BigInt c;
   c.len = 0;

   for(int i = 0, vaiUm = 0; vaiUm || i < max(len, b.len); i++)
   {
      int x = vaiUm;
      if(i < len) x += num[i];
      if(i < b.len) x += b.num[i];
      c.num[c.len++] = x % 10;
      vaiUm = x / 10;
   }
   return c;
}
```

SUBTRAÇÃO

```
BigInt BigInt::subtrai(BigInt b)
{
   BigInt c;
   c.len = 0;

   for(int i = 0; i < max(len, b.len); i++)
   {
      int x = 0;
      if(i < len) x += num[i];
      if(i < b.len)
      {
         if(num[i] >= b.num[i])
            x -= b.num[i];
         else {
            x += 10 - b.num[i];
            num[i+1]--;
         }
      }

            c.num[c.len++] = x;
   }
   return c;
}
```

3ª Questão

3ª Questão Criar uma classe que represente um **triangulo retângulo**.

Criar três membros de dados inteiros para representar o tamanho dos lados, e verificar se esses dados realmente formam um triangulo.

Implementar uma função membro que imprima o valor dos lados de todos os possíveis triângulos retângulos formados por três números inteiros menores que 50.

3ª questão
Exemplo de Solução

```
#ifndef TRIANGULO_H
#define TRIANGULO_H

class TrianguloRet{
   private:
      int m_a, m_b, m_c;

   public:
      TrianguloRet() {};
      TrianguloRet(int, int, int);
      ~TrianguloRet() {};

      void todos100();
};

#endif
```

```cpp
#include <iostream>
#include <math.h>
#include "triangulo.h"

using namespace std;

void TrianguloRet::todos100(){
        for(int i=1;i<=100;i++){
                for(int j=1;j<=100;j++){
                        for(int k=1;k<=100;k++){
                                if(i==sqrt(j*j + k*k)){
                                        cout << i << " " << j << " " << k << endl;
                                }
                        }
                }
        }
}

TrianguloRet::TrianguloRet(int a, int b, int c){
        if(((a*a) == (b*b + c*c)) ||
                        ((b*b) == (a*a + c*c)) ||
                                ((c*c) == (a*a + b*b)) ){
                m_a = a;
                m_b = b;
                m_c = c;
        }else{
                cout << "Nao e triangulo. Valor padrao a=5, b=4 , c=3"<<endl;
                m_a = 5;
                m_b = 4;
                m_c = 3;
        }
}
```

Main

```
#include "triangulo.h"

int main(){
   TrianguloRet triangulo(1,10,1);
   triangulo.todos100();
}
```