



# ECOP13-Lab7

---

Guia de Laboratório  
Prof. André Bernardi  
[andrebernardi@unifei.edu.br](mailto:andrebernardi@unifei.edu.br)



# 7º Laboratório ECOP13

## 14 de outubro 2022

---



# 1ª Questão

---

Crie um programa principal para testar a hierarquia Politico/Presidente/ Governador/Prefeito, da tarefa 2 do laboratório 6 com as seguintes características.

- Utilize um vetor de ponteiros para a classe base de modo a representar os diversos elementos a serem criados.
- Permita que o usuário escolha qual tipo de objeto será criado através de um menu de opções do tipo: 1-Presidente, 2-Governador, 3-Prefeito.
- Utilize um loop para imprimir todos os objetos criados



## 2ª Questão

---

Alterar a hierarquia criada no ex2 do laboratório 6 de modo a permitir o uso da propriedade do polimorfismo.

Modifique o tipo da função Imprime() para que ela seja virtual e execute novamente o programa para analisar o resultado.

# 1ª e 2ª questão

## Exemplo de Solução



```
#ifndef POLITICO_H
#define POLITICO_H

#include <string>
#include <iostream>

using namespace std;

// função de apoio para as classes
int menu();

class Politico{
protected:
    string nome, partido, numero;
public:
    Politico(string n="", string p="", string nu=""): nome{n},
        partido{p}, numero{nu}
    {
        cout << "Construindo Politico!" << endl;
    }

    virtual void read();
    virtual void imprime();

    friend ostream& operator<<(ostream&, Politico&);
    friend istream& operator>>(istream&, Politico&);

    virtual ~Politico( ) { cout << "Destroi Politico!" << endl << endl; }
};
```



```
class Presidente: public Politico{
protected:
    string pais;
public:
    Presidente(string nome="", string partido="", string numero="",
               string p=""): Politico{nome, partido, numero}, pais{p}
    {
        cout << "Construindo Presidente!" << endl;
    }

    void read();
    void imprime();

    ~Presidente() { cout << "Destroi Presidente!" << endl; }
};
```

```
class Governador: public Presidente{
protected:
    string estado;
public:
    Governador(string nome="", string partido="", string numero="",
               string pais="", string e=""): Presidente{nome, partido, numero, pais},
               estado{e}
    {
        cout << "Construindo Governador!" << endl;
    }

    void read();
    void imprime();

    ~Governador() { cout << "Destroi Governador!" << endl; }
};
```



```
class Prefeito: public Governador{
protected:
    string cidade;
public:
    Prefeito(string nome="", string partido="", string numero="",
            string pais="", string estado="", string c=""):
        Governador(nome, partido, numero, pais, estado), cidade{c}
    {
        cout << "Construindo Prefeito!" << endl << endl;
    }

    void read();
    void imprime();

    ~Prefeito() { cout << "Destroi Prefeito!" << endl; }
};
#endif // POLITICO_H
```



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

using namespace std;

ostream& operator<<(ostream& out, Politico& p)
{
    p.imprime();
    return out;
}

istream& operator>>(istream& in, Politico& p)
{
    p.read();
    return in;
}

void Politico::imprime()
{
    cout << "Numero:      " << numero << endl;
    cout << "Nome:          " << nome << endl;
    cout << "Partido:       " << partido << endl;
}
```





```
void Politico::read()
{
    cout << "Digite o nome: ";
    //getline(cin, nome);
    cin >> nome;

    cout << "Digite o partido: ";
    //getline(cin, partido);
    cin >> partido;

    cout << "Digite numero do candidato: ";
    cin >> numero;
}

void Presidente::imprime()
{
    Politico::imprime();
    cout << "Pais:          " << pais << endl;
}

void Presidente::read()
{
    Politico::read();
    cout << "Digite o pais: ";
    cin >> pais;
}
```



```
void Governador::imprime()
{
    Presidente::imprime();
    cout << "Estado:      " << estado << endl;
}

void Governador::read()
{
    Presidente::read();
    cout << "Digite o estado: ";
    cin >> estado;
}

void Prefeito::imprime()
{
    Governador::imprime();
    cout << "Cidade:      " << cidade << endl << endl;
}

void Prefeito::read()
{
    Governador::read();
    cout << "Digite a cidade: ";
    cin >> cidade;
}
```

```
//função de apoio das classes
int menu()
{
    int op;
    cout << "1 - Presidente" << endl;
    cout << "2 - Governador" << endl;
    cout << "3 - Prefeito  " << endl;
    cout << "0 - Sair      " << endl;

    cout << "Digite opcao desejada para inserir: ";

    while(cin >> op){
        if(op == 0 || op == 1 || op == 2 || op == 3)
            return op;

        cout << "Digite uma das opcoes disponiveis" << endl;
    }
}
```



# Main



CPP

```
#include <string>
#include <iostream>
#include "politico.h"
using namespace std;

#define MAXP 100000

int main(){

    int qtd;
    cout << "Informe quantidade politicos: ";
    cin >> qtd; cout << endl;

    if(qtd < 0 || qtd >= MAXP)
        qtd = 5; //limita em 5 caso usuário erre na quantidade

    Politico* vet[ qtd ];

    cout << "Escolha " << qtd
        << " vezes qual o tipo que deseja criar" << endl;
```

```
for(int j = 0; j < qtd; j++){
    cout << endl;
    int op = menu();

    if(op == 0)
        return 0;

    switch( op ){
        case 1:
            vet[ j ] = new Presidente; cout << endl;
            cin >> *vet[ j ];
            break;
        case 2:
            vet[ j ] = new Governador; cout << endl;
            cin >> *vet[ j ];
            break;
        case 3:
            vet[ j ] = new Prefeito; cout << endl;
            cin >> *vet[ j ];
            break;
    }
}
```

```
cout << endl << "### MOSTRAR VETOR! ###" << endl << endl;
for(int i = 0; i < qtd; i++)
    cout << *vet[ i ] << endl;

cout << endl << "#### FIM ####" << endl << endl;

//limpar memória.
for(int i = 0; i < qtd; i++)
    delete vet[ i ]; // chama o destrutor que deve ser virtual.

return 0;
}
```





## 3ª Questão

---

Crie um programa principal para testar a hierarquia **Ponto/Circulo/Cilindro**, do ex3 do laboratório 6 com as seguintes características.

- Utilize um vetor de ponteiros para a classe base de modo a representar os diversos elementos a serem criados.
- Permita que o usuário escolha qual tipo de objeto será criado através de um menu de opções do tipo: 1-Ponto, 2-Circulo, 3-Cilindro.
- Utilize um loop para imprimir todos os objetos criados



# 3ª e 4ª questão

## Exemplo de Solução – Classe Ponto

```
#include <iostream>
using namespace std;
#define PI 3.1415
#ifndef CIRCULO_H
#define CIRCULO_H

int menu();

class Ponto
{
protected:
    double x, y;
public:
    Ponto(double x = 0, double y = 0): x{x}, y{y} { }
    virtual ~Ponto() {}

    virtual void read()          { cin >> x >> y; }
    virtual void print() const   { cout << "C(" << x << "," << y << ")"; }


    virtual double area()       { return 0; }
    virtual double volume()     { return 0; }

    friend istream& operator>>(istream& input, Ponto& in);
    friend ostream& operator<<(ostream& output, const Ponto& out);
};
```



## 3ª e 4ª questão

# Exemplo de Solução – Classe Circulo



```
class Circulo: public Ponto
{
protected:
    double raio;
public:
    Circulo(double x = 0, double y = 0, double r = 0): Ponto{x, y}, raio {r} {}
    ~Circulo() {}


    void read()          { cin >> x >> y >> raio; }
    void print() const { Ponto::print();  cout << " RAI0 = " << raio; }

    double area()      { return PI*raio*raio; }
    double volume()    { return 0; }

};
```

## 3ª e 4ª questão

# Exemplo de Solução – Classe Cilindro



```
class Cilindro: public Circulo
{
protected:
    double altura;
public:
    Cilindro(double x = 0, double y = 0, double raio = 0, double a = 0):
Circulo{x, y, raio}, altura{a} {}
    ~Cilindro() {}

    void read()          { cin >> x >> y >> raio >> altura; }
    void print() const { Circulo::print();  cout << " ALTURA = " << altura; }

    double area()      { return (2*Circulo::area())+ 2*PI*raio*altura ; }
    double volume() { return (Circulo::area())*altura; }

};

#endif // CIRCULO_H
```



```
#include <iostream>
#include "circulo.h"

using namespace std;

istream& operator>>(istream& input, Ponto& in){
    in.read();
    return input;
}

ostream& operator<<(ostream& output, const Ponto& out){
    out.print();
    return output;
}
```

```
int menu() {
    int op;
    cout << "1 - Ponto    " << endl;
    cout << "2 - Circulo  " << endl;
    cout << "3 - Cilindro" << endl;
    cout << "0 - Sair      " << endl;

    cout << "Digite opcao desejada para inserir: ";

    while(cin >> op) {
        if(op == 0 || op == 1 || op == 2 || op == 3)
            return op;

        cout << "Digite uma das opcoes disponiveis" << endl;
    }
}
```

# Main



```
#include <iostream>
#include "circulo.h"

using namespace std;

#define PI 3.1415
#define MAXP 100000

int main(){

    int qtd;
    cout << "Informe quantidade formas: ";
    cin >> qtd; cout << endl;

    if(qtd < 0 || qtd >= MAXP)
        qtd = 5;

    Ponto* vet[ MAXP ];

    cout << "Escolha " << qtd
        << " vezes qual o tipo que deseja criar" << endl;
```

```

for(int j = 0; j < qtd; j++){

    cout << endl;
    int op = menu();
    if(op == 0)
        return 0;

    switch( op ){
        case 1:
            vet[ j ] = new Ponto; cout << endl;
            cin >> *vet[ j ];
            break;
        case 2:
            vet[ j ] = new Circulo; cout << endl;
            cin >> *vet[ j ];
            break;
        case 3:
            vet[ j ] = new Cilindro; cout << endl;
            cin >> *vet[ j ];
            break;
    }
}

```

```

cout << endl << "### MOSTRAR VETOR! ###" << endl << endl;
for(int i = 0; i < qtd; i++)
    cout << *vet[ i ] << endl;

cout << endl << "#### FIM ####" << endl << endl;

//limpar memória.
for(int i = 0; i < qtd; i++)
    delete vet[ i ]; // chama o destrutor que deve ser virtual.

return 0;
}

```