Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Algoritmos e Estrutura de dados 1 - 2025/1
Professor Lucas Malacarne Astore

Lista 14 - Herança

1. **Exercício - Ponto**

   (a) Definir uma classe que represente um Ponto.

   (b) Criar construtores para instanciar um ponto, com e sem parâmetros.

   (c) Criar métodos públicos para:

   - Definir a posição do ponto;
   - Obter a posição do ponto;
   - Calcular a distância entre dois pontos, dado outro objeto da classe ponto;
   - Calcular a distância entre dois pontos, dadas as coordenadas de outro ponto, como um par de números.

   (d) Criar a classe Circulo derivada da classe Ponto.

   (e) Criar construtores para instanciar um círculo, com e sem parâmetros.

   (f) Criar a classe Esfera derivada da classe Circulo. Esta classe deve representar uma esfera e conter um método adicional para calcular o volume da esfera. Use a fórmula:

$$\text{Volume} = \frac{4}{3}\pi r^3$$

   (g) Criar um programa principal para testar as classes criadas.

2. **Gerenciamento de Dados de Carros com Structs**

Desenvolva um programa em C++ para gerenciar os dados de um grupo de carros. Cada carro é representado por uma estrutura que contém as seguintes informações:

- Modelo do carro
- Marca do carro
- Ano de fabricação do carro
- Preço do carro
- Data da venda do carro (outra estrutura contendo dia, mês e ano)

a) Crie uma estrutura chamada `Data` contendo três campos: `dia`, `mês` e `ano`.

b) Crie uma estrutura chamada `Carro` para representar um carro, que deve conter os campos mencionados acima: modelo, marca, ano de fabricação, preço e data da venda.

c) Suponha que as estruturas criadas anteriormente tenham escopo global e possam ser usadas no programa principal e nas demais funções do código. Escreva uma função chamada `carroMaisCaro` que receba como parâmetro uma lista contendo os dados de n carros (structs do tipo `Carro`) e mostre na tela o preço e a data da venda do carro mais caro da lista. A função deve obedecer à seguinte definição:

```
void carroMaisCaro(Carro lista[], int n);
```

O programa principal que testa as estruturas e a função criadas está na página seguinte.

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    int n;
    cout << "Digite o numero de carros: ";
    cin >> n;

    Carro* lista = new Carro[n];

    for (int i = 0; i < n; i++) {
        cout << "\nDigite dados do carro " << i + 1 << ":" << endl;

        cout << "Modelo: ";
        cin >> ws; // Limpa o buffer
        getline(cin, lista[i].modelo);

        cout << "Marca: ";
        getline(cin, lista[i].marca);

        cout << "Ano de fabricacao: ";
        cin >> lista[i].anoFabricacao;

        cout << "Preco: ";
        cin >> lista[i].preco;

        cout << "Data da venda (dia mes ano): ";

        cin >> lista[i].dataVenda.dia
            >> lista[i].dataVenda.mes
            >> lista[i].dataVenda.ano;
    }

    carroMaisCaro(lista, n);

    delete[] lista;
    return 0;
}


}
```

3. **Cálculo de Salário de Vendedores**

Dada a classe `Empregado` com os seguintes atributos e métodos:

```
class Empregado {
private:
    String nome;
    double salarioBase;
    double imposto;

// Construtores padrao e com parametros
public:
    Empregado() : nome(""), salarioBase(0.0), imposto(0.0) {}

    Empregado(String nome, double salarioBase, double imposto) {
        this->nome = nome;
        this->salarioBase = salarioBase;
        this->imposto = imposto;
    }

    // Metodos getters e setters
    String getNome() {
        return nome;
    }

    void setNome(String nome) {
        this->nome = nome;
    }

    double getSalarioBase() {
        return salarioBase;
    }

    void setSalarioBase(double salarioBase) {
        this->salarioBase = salarioBase;
    }

    double getImposto() {
        return imposto;
    }

    void setImposto(double imposto) {
        this->imposto = imposto;
    }
};
```

a) Implemente a classe `Vendedor` como subclasse da classe `Empregado`. Um determinado vendedor tem, além dos atributos da classe `Empregado`, os seguintes atributos adicionais:

- `double valorVendas` (correspondente ao valor monetário dos artigos vendidos)
- `double comissao` (porcentagem do `valorVendas` que será adicionada ao salário base do vendedor)

Implemente os construtores e os métodos para manipular os atributos (`get` e `set`) da classe `Vendedor`.

b) Na classe `Vendedor`, implemente um método `calcularSalario` que retorne o salário líquido do vendedor, incluindo a comissão no cálculo do salário e deduzindo os impostos. O cálculo do salário líquido pode ser representado pela fórmula:

$$salarioLiquido = (salarioBase + (comissao \times valorVendas)) \times (1 - imposto)$$

O programa principal que testa a classe `Vendedor` e o método `calcularSalario` deve ser implementado como segue:

```
int main() {
    // Vendedor 1: Mariano
    Vendedor vendedor1("Mariano", 1000.00, 0.14, 80000.00, 0.05);

    // Vendedor 2: Joana
    Vendedor vendedor2("Joana", 1500.00, 0.12, 120000.00, 0.06);

    cout << "Vendedor 1: " << vendedor1.getNome() << endl;
    cout << "Salario Liquido: R$ " << vendedor1.calcularSalario() << endl;

    cout << "\nVendedor 2: " << vendedor2.getNome() << endl;
    cout << "Salario Liquido: R$ " << vendedor2.calcularSalario() << endl;

    return 0;
}
```