Universidade Federal de Mato Grosso

Bacharelado em Ciência da Computação

Semestre: 2024/1

Programação 2

Lista 5

Exercício 1: Hierarquia de Animais

Enunciado

Implemente uma hierarquia de classes para representar animais. Crie uma classe base chamada `Animal` com um método virtual `emitirSom()`. Derive classes `Cachorro`, `Gato` e `Vaca`, cada uma implementando o método `emitirSom()` de forma apropriada. Crie um vetor de ponteiros para `Animal` e adicione instâncias das classes derivadas. Percorra o vetor chamando `emitirSom()` para cada animal.

Main e Saída Esperada

Listing 1 – Main do Exercício 1

```
int main() {
    Cachorro cachorro;
    Gato gato;
    Vaca vaca;

    vector<Animal*> animais;
    animais.push_back(&cachorro);
    animais.push_back(&gato);
    animais.push_back(&vaca);

    for (Animal* animal : animais) {
        animal->emitirSom();
    }

    return 0;
}

Saída esperada:
Au Au
Miau
Muuu
```

Exercício 2: Sistema de Pagamento

Enunciado

Implemente um sistema de pagamento utilizando herança e polimorfismo. Crie uma classe base chamada `Empregado` com métodos virtuais para calcular o salário mensal e imprimir detalhes do empregado. Derive classes `EmpregadoAssalariado` e `EmpregadoHorista`. Crie um vetor de ponteiros para `Empregado` e adicione instâncias das classes derivadas. Percorra o vetor chamando os métodos apropriados para cada empregado.

Main e Saída Esperada

Listing 2 – Main do Exercício 2

```
int main() {
    vector<shared_ptr<Empregado>> empregados;
    empregados.push_back(make_shared<EmpregadoAssalariado>(3000.00));
    empregados.push_back(make_shared<EmpregadoHorista>(160, 25.00));

    for (const auto& empregado : empregados) {
        empregado->imprimirDetalhes();
        cout << "Salario Mensal: " << empregado->getSalarioMensal() <<
            endl;
    }

    return 0;
}

Saída esperada:
Empregado Assalariado, Salário Mensal: 3000
Salário Mensal: 3000
Empregado Horista, Horas Trabalhadas: 160, Taxa por Hora: 25
Salário Mensal: 4000
```

Exercício 3: Formas Geométricas

Enunciado

Implemente uma hierarquia de classes para representar formas geométricas. Crie uma classe base chamada `Forma` com um método virtual `area()`. Derive classes `Circulo`, `Retangulo` e `Triangulo`, cada uma implementando o método `area()` de forma apropriada. Crie um vetor de ponteiros para `Forma` e adicione instâncias das classes derivadas. Percorra o vetor chamando `area()` para cada forma.

Main e Saída Esperada

Listing 3 – Main do Exercício 3

```
int main() {
    vector<Forma*> formas;
    formas.push_back(new Circulo(5.0));
    formas.push_back(new Retangulo(4.0, 6.0));
    formas.push_back(new Triangulo(4.0, 5.0));

    for (Forma* forma : formas) {
        cout << "Area: " << forma->area() << endl;
    }

    for (Forma* forma : formas) {
        delete forma;
    }

    return 0;
}
```

```
Saída esperada:
Área: 78.5398
Área: 24
Área: 10
```

Exercício 4: Biblioteca de Mídia

Enunciado

Implemente uma hierarquia de classes para representar mídias. Crie uma classe base chamada `Midia` com métodos virtuais `exibirInfo()` e `reproduzir()`. Derive classes `Livro`, `CD` e `DVD`, cada uma implementando os métodos de forma apropriada. Crie um vetor de ponteiros para `Midia` e adicione instâncias das classes derivadas. Percorra o vetor chamando os métodos apropriados para cada mídia.

Exercício 5: Sistema de Transporte

Enunciado

Implemente uma hierarquia de classes para representar meios de transporte. Crie uma classe base chamada `Transporte` com métodos virtuais `iniciar()` e `parar()`. Derive classes `Carro`, `Bicicleta` e `Aviao`, cada uma implementando os métodos de forma apropriada. Crie um vetor de ponteiros para `Transporte` e adicione instâncias das classes derivadas. Percorra o vetor chamando os métodos apropriados para cada meio de transporte.