**FEMA-IMESA**

**Curso:** Bacharelado em Ciência da Computação **Coordenadoria:** Informática

**Disciplina:** Linguagem de Programação

**Professor Responsável:** Dr.Almir R. Camolesi

**Lista de Exercícios:** Implementações de Classes

1) Desenvolva utilizando os conceitos de C++, as seguintes classes

O mercado mundial de bebidas tem crescido fortemente em relação a vinhos. Existem pesquisas que o consumo de um bom vinho, consumido em doses adequadas faz bem a saúde. Sabe-se que o sabor de um vinho é constituído pelo sabor da uva que é utilizada em sua fabricação. A Federação dos Produtores de Vinho do Brasil solicitou que sejam elaboradas classes em C++ para controlar os vinhos produzidos, o Pais de Origem e as respectivas uvas utilizadas.

|  |  |
| --- | --- |
| Classe Uva private: cod: inteiro; nome: string; cor: string; paisOrigem: Pais; público: Uva();  Uva (int, string, string, string, string);  …… | Classe Vinho: herda Uva privado: cod: inteiro; fabricante: Fabricante nome: string; valor: real; público: Vinho(); |

Utilizando-se dos conceitos de orientação a objetos apresentados em aula, etc.. Com base nas definições acima, desenvolva as classes Fabricante, País, Uva e Vinho. Para cada Classe desenvolva:

1. Métodos construtores: padrão e com argumentos;
2. Modificadores (get e set)
3. Métodos polimórficos para alterar o valor de um vinho. Um método deve receber o valor do percentual de aumento e realizar tal tarefa. O segundo método deve assumir um aumento de 5% no valor, caso não seja informado nenhum parâmetro;
4. função *main ( )* para demonstrar a utilização das classes e métodos definidos.

Obs: ao desenvolver as classes utilize os conceitos de métodos in line e proteção de atributos e variáveis const.

1. Implemente todas as funções membros das seguintes classes:

|  |  |
| --- | --- |
| a) Classe Pessoas class Pessoa  { public:  Pessoa ();  Pessoa (string nome, int idade);  void lerPessoa(); void impPessoa(); string retNome () const; int retIdade() const; void setNome (string); void setIdade(int); private:  string nome;  int idade;  }; | b) Classe Funcionário class Funcionario  { public:  Funcionario ();  Funcionario (string funcNome, float funSal): void lerFunc (); void impFunc();  void setSalario (float novoSal);  void setNomeFunc(); float ret\_Salario () const; string get\_NomeFunc () const; private:  Pessoa dadosFunc; float salario;  }; |

1. Implemente uma classe *Address* (*Endereço*). Um endereço possui uma rua, um número e um número de apartamento opcional, uma cidade, um estado e um código postal. Forneça dois construtores um com número de apartamento e outro sem. Forneça um método *print* que imprima o endereço com a rua em uma linha e a cidade, o estado e o código postal na linha seguinte. Forneça uma função no programa principal, *comes\_before* (*vem\_antes*) que recebe por parâmetro dois dois objetos *Address*, e verifica qual endereço vem primeiro, ou seja, qual possui código postal menor. Utilize passagem de objetos por ponteiro e constantes para proteger os objetos passados para a função *comes\_before*.

1. Implemente uma classe Account (Conta). Uma conta possui um saldo, funções que depositam e retiram dinheiro e uma função para informar o saldo atual. A função retira dinheiro deve cobrar uma multa de R$5,00, caso seja feita uma tentativa de retirar mais dinheiro do que possível na conta.

1. ☺O que são funções membros, e como ele difere de uma função não membro?

R: Os modelos da função de membro são funções de modelo que são membros de uma classe ou de um modelo de classe, ela se difere de uma não membro porque está diretamente ligada a classe.

1. ☺O que acontece se você esquecer de usar const em um método de acesso? O que acontece se você acidentalmente coloca const em uma função modificadora?

R: Este método tem a possibilidade de causar alterações indevidas no código, se const for usado em uma função modificadora ele impedirá ela de fazer modificações.

1. ☺O quê é um construtor? O que é um construtor default? Qual é a conseqüência se uma classe não possui um construtor default?

R: É o método responsável pela instanciação dos objetos, ele pode ter dois tipos que são:

-Padrão: Neste caso, o objeto é criado com atributos zerados, quando ele é chamado pelo programa, não é necessário a passagem de argumentos.

-Parametrizado: No construtor parametrizado, o objeto irá receber valores passados na chamada do método, através da passagem de argumentos, nesse caso os atributos do objeto em questão serão definidos de acordo com os dados enviados pelo usuário.

Se uma classe não possuir construtor padrão definido, ela criará um automaticamente e irá definir valores 0 ou nulos, a consequência é que não haverá controle sobre o construtor padrão.

1. ☺O que é um método in line? Dê exemplos.

R: É um método que foi escrito em uma linha, exemplos:

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

int getId const() {return this->id;}

1. ☺Demonstre o procedimento para iniciação de um construtor in line.

R: Aluno(int RA, string nome, float nota): RA(RA), nome(nome), nota(nota) {}

1. ☺O que é a interface de uma classe? O que é a implementação de uma classe? **(1,5)**

R: É uma classe sem nenhuma implementação, todas as operações são abstratas, os métodos têm apenas a sua assinatura, o relacionamento de realização indica as classes que implementam a interface. Implementação é a codificação de uma classe, é o que será passado para execução na máquina.

1. ☺Você pode ler e escrever o atributo *salário* da classe Empregado com o método de acesso *ObeterSalario( )* e o método de alteração de dados *AlteraSalario( )*. Cada atributo de uma classe deve ter métodos de acesso e de alteração a dados. Explique por quê sim e por quê não?

R: Na maioria dos casos, esses métodos são necessários para que seja possível a manipulação de determinado dado, pois normalmente os atributos de uma classe são protegidos, de modo a serem acessáveis somente através de métodos, assim o único jeito de trabalhar com essas informações é através de métodos getters e setters, com exceção de alguns programas que utilizam outras maneiras para manipular os dados, excluindo a necessidade desses métodos.

1. ☺Campos de dados são ocultos na seção privativa de uma classe, mas eles não estão muito bem escondidos. Qualquer um pode ler os dados da seção privativa. Explique até que ponto a palavra chave *private* esconde os atributos privativos de uma classe.

R: A palavra private encapsula os dados, deixando ela disponível apenas para os métodos da própria classe, mas vale lembrar que esses métodos podem ser chamados por qualquer um, portanto ela não está tão escondida assim.

1. ☺O que é sobrecarga de operadores? Qual a importância deste conceito relacionado à programação orientada a objetos? Dê exemplos.

R: Sobrecarga é quando existem duas ou mais funções que compartilham o mesmo nome, mas que recebem argumentos diferentes, e realizam atividades distintas. A sobrecarga é importante para manipular dados de maneira parecida, porém com parâmetros diferenciados.

Por exemplo:

void Cat::sound(){

cout << "Meow ";

}

void Cat::sound(int num){

for(int i=0;i<num;i++){

cout << "Meow ";

}

}

Nesse exemplo, quando a função sound for chamada, se não forem passados argumentos, um “Meow” será reproduzido, caso seja passado um determinado número, ele repetirá o “Meow” conforme o número passado.

1. Defina uma classe *Produto*. Um produto possui Código, descrição, valor e fornecedor. O atributo fornecedor pertence à classe *Fornecedor* que possui os seguintes atributos: código, nome, endereço e telefone. Supunha que a classe *fornecedor* já esteja implementada, implemente apenas a classe *Produto*
   * Forneça um construtor para iniciar os valores numéricos com zero e os textuais com “”.
   * Forneça uma função *imprime* que imprima os dados de um produto e o nome e o endereço de um fornecedor.
   * Forneça uma função que realize a leitura dos atributos da classe *Produto*;
   * Projete dois métodos para atualizar valor do produto: um método que ao ser chamado soma automaticamente 20% ao valor do produto e um outro método que receba por parâmetro o percentual de aumento.
   * Desenvolva um programa principal para demonstrar a utilização da classe.

1. ☺O que são construtores e destrutores? Qual a importância dos mesmos para o estilo de programação orientado a objetos?

R: Construtores são métodos para instanciação e definição de valores para um objeto, destrutores por sua vez são métodos para limpeza de um objeto, apagando seus valores e liberando memória. Métodos construtores são importantes na instanciação e para que haja validação ou algum processamento prévio na leitura de dados, e os destrutores são importantes para liberar espaço em memória.

1. ☺Em relação a objetos o que é escopo e tempo de vida? Dê exemplos.

R: -Escopo: Este é um conceito de tempo de compilação. Escopo (pelo menos o estático) existe no texto do código fonte, você o descobre lendo o código. Ele é um indicador do tempo de vida de uma vinculação de um nome com um valor. Em geral isto se refere à variáveis. Ele se refere em que partes do código a variável é visível/acessível.

-Tempo de vida: O “tempo de vida” é o período durante a execução de um programa em que uma variável ou uma função existe. A duração de armazenamento de identificador determina seu tempo de vida.

Boa Diversão!!!!