PUC Minas

Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática (ICEI)
Engenharia de Computação / Engenharia de Software
Disciplina: Algoritmos e Estruturas de Dados I

Lista de Exercícios 10

Informações sobre cópias

As questões são individuais. Em caso de cópias de trabalho a pontuação será zero para os autores originais e copiadores. Não serão aceitas justificativas como: "Fizemos o trabalho juntos, por isso estão idênticos".

Parte A – Programação Orientada a Objetos - Exercícios para serem entregues

Resolva os exercícios a seguir e entregue pelo CANVAS. Cada exercício deve conter um arquivo no **formato .CPP.**

Conceitos de Classe e Encapsulamento

A01. Classe Básica - Crie uma classe chamada Ingrediente com os atributos privados nome (do tipo char[]), quantidade (do tipo double) e unidade (do tipo char[], representando unidades como kg, litros, unidades). Implemente métodos públicos para definir e obter os valores de nome, quantidade e unidade, além de um método para imprimir as informações do ingrediente.

Função *main*: crie um objeto da classe Ingrediente, defina os valores dos atributos e imprima-os.

Encapsulamento e Construtores

A02. Modifique a classe Ingrediente para incluir um construtor que receba nome, quantidade e unidade como parâmetros, além de construtores padrão e de cópia, e um destrutor. Adicione verificações nos métodos setters para garantir que a quantidade não seja negativa e que unidade seja uma unidade válida (como kg, litros, unidades).

Função *main*: teste a classe criando objetos de diferentes maneiras, usando diferentes construtores, e imprimindo suas informações.

Conceito de Herança

A03. Herança Básica - Crie uma classe base chamada Produto com os atributos nome (do tipo char[]) e preco (do tipo double). Implemente métodos públicos para definir e obter os valores de nome e preco, e um método para imprimir as informações do produto. Em seguida, crie duas classes derivadas: Bebida e Alimento, que herdam de Produto. A classe Bebida deve ter um atributo adicional volume (do tipo double) e a classe Alimento deve ter um atributo adicional peso (do tipo double). Implemente métodos para definir e obter os valores de volume e peso nas respectivas classes derivadas.

Função *main*: crie objetos das classes Bebida e Alimento, defina os valores dos atributos e imprima-os.

Conceito de Polimorfismo

A04. Continue com a estrutura do Exercício 3. Adicione um método virtual descricao na classe Produto que imprima uma descrição genérica do produto. Sobrescreva o método descricao nas classes derivadas Bebida e Alimento para imprimir descrições específicas (por exemplo, "Esta é uma bebida com volume de X litros" para Bebida e "Este é um alimento com peso de Y kg" para Alimento). Crie uma função que receba um ponteiro para Produto e invoque o método descricao. Em seguida, crie um vetor de ponteiros para Produto e adicione instâncias de Bebida e Alimento.
Função *main*: use a função criada para chamar o método descricao para cada produto no vetor.

Encapsulamento e Polimorfismo

A05. Crie uma classe Receita que contenha uma lista de Ingrediente. Implemente métodos para adicionar um ingrediente à receita, remover um ingrediente da receita e listar todos os ingredientes da receita. Em seguida, crie uma classe derivada ReceitaEspecial que adicione um método para calcular o custo total da receita com base nos preços dos ingredientes.

Função *main*: teste a classe ReceitaEspecial criando uma receita, adicionando ingredientes e calculando o custo total.

Parte B – Tratamento de Exceção (e POO) - Exercícios para serem entregues

B01. Explique detalhadamente, em texto, os comandos *try*, *catch* e *throw*.

B02. Explique a função da cláusula *throw* na assinatura do método abaixo.

void imprime_informacoes(int vetor[], int tamanho) throw(out_of_range);

B03. Explique o que é hierarquia de exceções. Dê um exemplo de utilização desse recurso.

B04. Pesquise e explique com suas palavras o comando *finally*, utilizado no contexto de tratamento de exceção.

B05. Escreva uma classe em C++ que gerencie jogos de loteria. A classe deve permitir a escolha de 6 números que variam de 1 a 60. A escolha de um número deve ser feita em um método especializado. Caso o escolhido seja repetido, inferior a 1 ou superior a 60 seja escolhido, o sistema deve gerar uma exceção nesse método, que deve ser propagada e tratada no

procedimento principal (main).

B06. Escreva uma classe em C++ que gerencie o acesso de um usuário ao sistema. O usuário deverá logar no sistema com o usuário "aluno" e a senha "1234". Caso o aluno tente entrar com um login diferente do informado, o sistema deverá lançar uma exceção, posteriormente tratada, indicando que o acesso não foi autorizado. Caso o usuário não informe o nome de usuário ou senha, o sistema também deve também lançar uma exceção, tratada pelo procedimento principal (main). A sua classe deve ter os atributos usuário e senha podendo ser carregados no construtor ou no método *logar*, que retornar um booleano indicando sucesso no login. Utilize o conceito de polimorfismo paramétrico (sobrecarga de método).

Parte C – Exercícios para Treino – Revisão (não precisam ser entregues)

C01. Escreva um programa para cadastrar clientes de uma loja. As informações necessárias são: nome, data de nascimento, endereço e telefone. Devem ser usada classes para a construção deste cadastro.

Utilize os conceitos de classe, atributos e método construtor. Crie o main, que declara um vetor de clientes e que permite ao usuário adicionar os clientes nesse vetor. O tamanho do vetor deverá ser definido pelo usuário.

C02. João gostaria de desenvolver um sistema de comparação de preços de um tipo específico de eletrodoméstico. Na primeira fase do seu software, gostaria de criar uma classe que contenha o nome da loja, telefone e preço de um eletrodoméstico. Em seguida, irá cadastrar um total de 15 registros e exibir estatísticas desses itens, de modo que possa verificar o melhor preço (menor valor), o preço médio e o preço máximo. Auxilie João a desenvolver esta classe e exiba as informações desejadas.

Utilize os conceitos de classe, atributos, métodos, método construtor. Proteja o acesso aos dados implementando encapsulamento por meio de métodos get e set. Crie o main, que declara um vetor de eletrodomésticos e que permite ao usuário adicionar nesse vetor informações das lojas e preço.

C03. Crie uma classe para representar uma conta corrente, com métodos para depositar uma quantia, sacar uma quantia e obter o saldo. Para cada saque será debitada também uma taxa de operação equivalente à 0.5% do valor sacado. Crie, em seguida, uma subclasse (herdada) da classe anterior para representar uma conta corrente de um cliente especial. Clientes especiais pagam taxas de operação de apenas 0.1% do valor sacado.

Utilize os conceitos de classe, atributos, métodos, encapsulamento (get/set), método construtor e herança. Crie um main com menu, que após o usuário definir o tipo de conta (comum ou especial), permitirá ao usuário depositar, sacar e mostrar saldo da cota instanciada.

C04. A empresa RH+ está criando um módulo para sua intranet, que deverá exibir os aniversariantes do dia. Para isso, solicitou à equipe de TI que construa uma sistemas para armazenar informações pessoais dos funcionários, como nome, dia de aniversário e mês de

aniversário. Após analisar o problema, ficou definido que deveriam ser construídas duas classes - uma que representasse a Pessoa e outra para gerenciamento dos aniversários. Crie um *main* que forneça, juntamente com as classes, funcionalidades para cadastro de pessoas e impressão de aniversariantes, por mês. Considere que a empresa possui no máximo 100 funcionários.

Utilize os conceitos de classe, atributos, métodos, encapsulamento (get/set), método construtor e relacionamento de classes com cardinalidade 1 para N (associação, agregação ou composição).

C05. Crie uma classe para representar uma pessoa, com os atributos privados de nome, idade e altura. Crie os métodos públicos necessários para sets e gets e também um método para imprimir os dados de uma pessoa. Ilustre o seu funcionamento em um programa *main.*

C06. Crie uma classe denominada Elevador para armazenar as informações de um elevador dentro de um prédio. A classe deve armazenar o andar atual (0=térreo), total de andares no prédio, excluindo o térreo, capacidade do elevador, e quantas pessoas estão presentes nele. A classe deve também disponibilizar os seguintes métodos:

- Inicializa: que deve receber como parâmetros: a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazios);
- Entra: para acrescentar uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
- Sai: para remover uma pessoa do elevador (só deve remover se houver alguém dentro dele);
- Sobe: para subir um andar (não deve subir se já estiver no último andar);
- Desce: para descer um andar (não deve descer se já estiver no térreo);
- Getters: métodos para obter cada um dos os dados armazenados.

Crie um main com menu acesso as funcionalidades inicializar, entrar, sair, subir, descer e exibir o status atual do elevador.

<u>Para os próximos exercícios seja criativo na construção do *main* e sua interação com as classes e usuário.</u>

C07. Um provedor de acesso à Internet mantém o seguinte cadastro de clientes: código do cliente, e-mail, número de horas de acesso, página (S-sim ou N-não). Elabore um algoritmo que calcule e mostre um relatório contendo o valor a pagar por cada cliente, sabendo-se que as primeiras 20 horas de acesso é R$35,00 e as horas que excederam tem o custo de R$2,50 por hora. Para os clientes que têm página, adicionar R$40,00. Inserir um conjunto de registros (máximo 500).

C08. Escreva uma classe em C++ para gerenciamento de mesas em restaurantes. O restaurante deve cadastrar a lotação máxima do estabelecimento, definida pelo número de mesas. Quando um cliente quiser entrar no estabelecimento, deve ser alocada uma mesa para ele (o nome do cliente deve ser alocado a uma mesa). Caso número de clientes seja superior ao número de mesas, o sistema deverá emitir uma exceção, indicando que o estabelecimento está cheio, tratada pelo procedimento principal (main).

C09. Altere o programa anterior para permitir a saída de clientes do restaurante. O primeiro cliente que entrou no restaurante deve ser o primeiro a sair. Caso número de clientes seja superior ao número de mesas, o sistema deverá emitir uma exceção, indicando que o estabelecimento está cheio.

Questões sobre contagem de Operações

C10. Seja o seguinte algoritmo:

```
int count = 0;
int i;
for (i=0; i<n; i++)
if (v[i] == 0)
count++;
```

calcule o número de operações:
- Declaração de Variáveis
- Atribuições

Calcule o número de operações em função de n:
- Comparação "menor que"
- Comparação "igual a"
- Acesso ao vetor
- Incremento no melhor caso
- Incremento no pior caso

C11. Seja o seguinte um algoritmo para verificar se um vetor contêm pelo menos dois valores duplicados (em qualquer lugar do vetor). Calcule o número de vezes que o comando if será executado no melhor e no pior caso. A resposta deve ser em função de n.

```
bool hasDuplicate(int *vet, int n){
  int i, j;
  bool duplicate = false;
  for (i = 0; i < n; i++){
    for (j = 0; j < n; j++){
      if (i != j && A[i] == A[j])
        return true;
    }
  }
  return false;
}
```

C12. Quantas operações de **soma** a função `func` realizara no **pior caso**? Forneça uma resposta em função de **n**.

```
#include <stdio.h>
 void func(int *v, int n)
 {
    int i, j;
    for(i = 2; i < n - 1; i++)
    {
         for(j = 0; j < n - 2; j++)
         {
             if( *(v + i) > *(v + j) )
             {
                 *(v + j) += *(v + i);
                 printf("%d %d %d %d \n", i, *(v+i), j, *(v+j));
             }
         }
     }
 }
```