Programação de Computadores 2
Lista 01 de Exercícios

Prof. Eduardo Cunha Campos

Data de entrega: 20/07/2021

Valor: 7,5 pontos

Aluno(a): _____

Matrícula: _____

Exercício 1. A fim de representar empregados em uma firma, crie uma classe chamada Empregado que inclui as três informações a seguir como atributos:

um primeiro nome,

um sobrenome, e

um salário mensal.

Sua classe deve ter um construtor que inicializa os três atributos. Forneça um método set e get para cada atributo. Se o salário mensal não for positivo, configure-o como 0.0. Escreva um aplicativo de teste que demonstra as capacidades da classe. Crie duas instâncias da classe e exiba o salário anual de cada instância. Então dê a cada empregado um aumento de 10% e exiba novamente o salário anual de cada empregado.

Exercício 2. Cria uma classe chamada *Complex* para representar números complexos e escreva um programa para testá-la.

1. Escolha uma representação para os números complexos, usando a forma retangular ou a forma polar.
2. Forneça três construtores que permitam que objetos dessa classe sejam inicializados ao serem alocados na memória:
    a) um construtor sem parâmetros que inicializa o objeto como zero

b) um construtor com um parâmetro representando a parte real; a parte imaginária será zero

c) um construtor com dois parâmetros representando as partes real e imaginária

3. Defina operações para obter a parte real, a parte imaginária, o módulo (valor absoluto) e o ângulo de um número complexo.
4. Forneça a operação para determinar o inverso aditivo de um número complexo.
5. Forneça as operações aritméticas básicas com números complexos: adição, subtração, multiplicação e divisão.
6. Forneça as operações relacionais que permitem comparar dois números complexos.
7. Defina a operação toString para converter um número complexo em string Utilize o formato (a; b), onde a é a parte real e b é a parte imaginária.
8. Escreva um aplicativo de teste que demonstra as capacidades da classe Complex.

Exercício 3. Crie uma classe para representar datas.

1. Represente uma data usando três atributos: o dia, o mês, e o ano.
2. Sua classe deve ter um construtor que inicializa os três atributos e verifica a validade dos valores fornecidos.
3. Forneça um construtor sem parâmetros que inicializa a data com a data atual fornecida pelo sistema operacional.
4. Forneça um método set um get para cada atributo.
5. Forneça o método toString para retornar uma representação da ata como string. Considere que a data deve ser formatada mostrando o dia, o mês e o ano separados por barra (/).
6. Forneça uma operação para avançar uma data para o dia seguinte.
7. Escreva um aplicativo de teste que demonstra as capacidades da classe.

Garanta que uma instância desta classe sempre esteja em um estado consistente.

Exercício 4. Grafo é uma estrutura de dados muito comum em computação, e os algoritmos sobre grafos são fundamentais para a área.

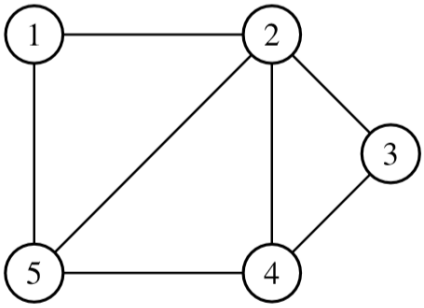
Um grafo G = (V; A) consiste em:

um conjunto finito de pontos V . Os elementos de V são chamados de vértices de G.

um conjunto finito A de pares não ordenados de V , que são chamados de arestas de G. Uma aresta a em A é um par não ordenado (v; w) de vértices v, w em V , que são chamados de extremidades de a.

Uma aresta a em A é chamada de incidente com um vértice v em V , se v for uma extremidade de a. Um vértice v em V diz-se vizinho de outro vértice w em V se existir uma aresta a em A incidente com v e w.

Um grafo pode ser representado por listas de adjacência ou por uma matriz de adjacência, como é ilustrado na figura 1.1.



(a) O grafo.

vértice	lista de adjacência
1	2, 5
2	1, 5
3	2, 4
4	2, 5, 3
5	4, 1, 2

(b) Listas de adjacência do grafo.

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c) Matriz de adjacência do grafo.

Figura 1.1: Um exemplo de grafo.

Escreva uma classe para representar grafos. Escolha entre a representação por listas de adjacência ou por matriz de adjacência. A classe deve oferecer uma operação para determinar se dois vértices são vizinhos, e outra operação para determinar a lista de todos os vértices que são vizinhos de um dado vértice. Considere que cada vértice é representado por um número inteiro. Escreva um aplicativo para testar a classe.