PUC Minas

PONTIFICIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Departamento de Ciência da Computação
Barreiro

Disciplina	Curso	Turno	Período
Algorítmos e Estruturas de Dados	Sistemas de Informação	Noite	2°

Professores
Daniel Capanema e Rafael Glater

Trabalho Prático 2: Data de Entrega: 30/04/2023 - Valor: 8 pontos

Classe

Neste Trabalho Prático sua tarefa é organizar as informações dos jogadores disponíveis para exibição ao usuário. Os dados foram obtidos a partir de uma busca no site Kaggle. Você deve ler, organizar e armazenar os dados de cada jogador em memória, utilizando as estruturas de dados em aula (Lista, Pilhas, Filas). Em seguida executar as operações descritas nos arquivos de entrada. Muito cuidado ao realizar o parser do texto. Fique atento a descrição dos dados que serão lidos e manipulados pelo seu sistema.

1. **Classe Jogadores em C#**: Crie uma classe *Jogadores*. Sua classe terá os atributos Nome (String), Foto (String), Nascimento (Data), Id(int) e Times(Array de inteiros). Ela terá também os métodos imprimir e ler. O método imprimir mostra a String `'id nome nascimento foto (times)'`, contendo todos os atributos da classe. O método ler deve efetuar a leitura dos atributos de um registro.

   A entrada padrão é composta por várias linhas e cada uma contém dados do jogador que devem ser tratados e armazenados em objetos da classe Jogador. A última linha da entrada contém FIM. A saída padrão também contém várias linhas, uma para cada registro contido na entrada padrão, conforme o exemplo abaixo:

   42373 Diego Alves 24/06/1985 https://tmssl.akamaized.net/images/portrait/header/42373-1543845950.jpg (614, 330)

2. **Pesquisa Sequencial em C#**: Faça a inserção de alguns objetos no final de uma Lista e, em seguida, faça algumas pesquisas sequenciais. A chave primária de pesquisa será o atributo nome do jogador. A entrada padrão é composta por duas partes onde a primeira é igual a entrada da primeira questão 1. As demais linhas correspondem a segunda parte. A segunda parte é composta por várias linhas. Cada uma possui um elemento que deve ser pesquisado na Lista. A última linha terá a palavra FIM. A saída padrão será composta por várias linhas contendo as palavras SIM/NAO para indicar se existe cada um dos elementos pesquisados.

3. **Pesquisa Binária em C#**: Repita a questão anterior, contudo, usando a Pesquisa Binária. A entrada e a saída padrão serão iguais às da questão anterior.

4. **Lista com Alocação Sequencial em C#**: Crie uma Lista de Jogadores baseada na lista de inteiros vista na sala de aula. A lista pode ter tamanho 20. Sua lista deve conter todos os atributos e métodos existentes na lista de inteiros, contudo, adaptados para a classe Jogador. De toda forma, lembre-se que, na verdade, temos uma lista de ponteiros e cada um deles aponta para um objeto Jogador. Neste exercício, faremos inserções, remoções e mostraremos os elementos de nossa lista. Os métodos de inserir e remover devem operar conforme descrito a seguir, respeitando parâmetros e retornos. Primeiro, o void inserirInicio(Jogador jogador) insere um objeto na primeira posição da Lista e remaneja os demais. Segundo, o void inserir(Jogador jogador, int

posição) insere um objeto na posição p da Lista, onde p < n e n é o número de objetos cadastrados. Em seguida, esse método remaneja os demais objetos. O void inserirFim(Jogador jogador) insere um objeto na última posição da Lista. O método removerInicio() remove e retorna o primeiro objeto cadastrado na Lista e remaneja os demais. O método remover(int posição) remove e retorna o objeto cadastrado na p-ésima posição da Lista e remaneja os demais. O método removerFim() remove e retorna o ultimo objeto cadastrado na Lista. A entrada padrão é composta por duas partes. A primeira é igual a entrada da primeira questão. As demais linhas correspondem a segunda parte. A primeira linha da segunda parte tem um número inteiro n indicando a quantidade de objetos a serem inseridos/removidos. Nas próximas n linhas, tem-se n comandos de inserção/remoção a serem processados neste exercício. Cada uma dessas linhas tem uma palavra de comando: II inserir no início, I* inserir em qualquer posição, IF inserir no fim, RI remover no início, R* remover em qualquer posição e RF remover no fim. No caso dos comandos de inserir, temos também o id do registro a ser inserido. No caso dos comandos de "em qualquer posição", temos também a posição. No Inserir, a posição fica imediatamente após a palavra de comando. A saída padrão tem uma linha para cada objeto removido sendo que essa informação será constituída pela palavra "(R)" e o atributo nome. No final, a saída mostra os atributos relativos a cada objeto cadastrado na lista após as operações de inserção e remoção.

5. **Pilha com Alocação Sequencial em C#**: Crie uma Pilha de Jogadores baseada na pilha de inteiros vista na sala de aula. A pilha pode ter tamanho 20. Neste exercício, faremos inserções, remoções e mostraremos os elementos de nossa pilha. A entrada e a saída padrão serão como as da questão anterior, contudo, teremos apenas os comandos I para inserir na pilha (empilhar) e R para remover (desempilhar).

6. **Fila Circular com Alocação Sequencial em C#**: Crie uma classe Fila Circular de Jogadores. Essa fila deve ter tamanho cinco. Em seguida, façaa um programa que leia vários registros e insira seus atributos na fila. Quando o programa tiver que inserir um objeto e a fila estiver cheia, antes, ele deve fazer uma remoção. A entrada e a saída padrão serão como as da questão anterior.

7. **Lista com Alocação dinâmica em C#**: Repita o exercício de lista, porém utilizando estruturas nativas (collections).

8. **Pilha com Alocação dinâmica em C#**: Repita o exercício de pilha, porém utilizando estruturas nativas (collections).

9. **Fila com Alocação dinâmica em C#**: Repita o exercício de fila, porém utilizando estruturas nativas (collections).