TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

INSTRUÇÕES GERAIS PARA O TRABALHO

1. Esta atividade poderá ser resolvida em grupo com no máximo 3 integrantes.

2. Caso você ache que falta algum detalhe nas especificações, você deverá fazer as suposições que julgar necessárias e escrevê-las no seu relatório. Pode acontecer também que a descrição dessa atividade contenha dados e/ou especificações supérfluas para sua solução. Utilize sua capacidade de julgamento para separar o supérfluo do necessário.

3. Como produtos da atividade serão gerados dois artefatos: códigos fontes da implementação e documentação da atividade.

4. Cada arquivo-fonte deve ter um cabeçalho constando as seguintes informações: nome(s) do(s) aluno(s), matrícula(s) e data.

5. O arquivo contendo a documentação da atividade (relatório) deve ser devidamente identificado com o(s) nome(s) e matrícula do(s) autor(es) do trabalho. O arquivo contendo o relatório deve, obrigatoriamente, estar no formado PDF.

6. Devem ser entregues os arquivos contendo os códigos-fontes e o arquivo contendo a documentação da atividade (relatório). Compacte todos os artefatos gerados num único arquivo no formato RAR.

7. Entregue apenas uma resolução por grupo.

8. O prazo final para entrega desta atividade é até 23:59:00 do dia 26/03/2021.

9. O envio é de total responsabilidade do aluno. Não serão aceitos trabalhos enviados fora do prazo estabelecido.

10. Trabalhos plagiados serão desconsiderados, sendo atribuída nota 0 (zero) a todos os envolvidos.

11. O valor desta atividade é 20 pontos.

---

1 INTRODUÇÃO

As estruturas de dados do tipo pilha e fila são consideradas listas especializadas por possuírem características próprias. Mas, apesar dessas características, ambas possuem operações como inserir um elemento, excluir um elemento, encontrar o maior, encontrar o menor, contar os elementos, entre outros.

As duas estruturas de dados, pilhas e filas, representam conjuntos de dados que estão organizados em ordem linear. Quando essas estruturas são representadas por arranjos, ou seja, quando é feita a utilização de vetores nas representações, tem-se o uso de

endereços contíguos de memória do computador e a ordem linear é determinada pelos índices dos vetores, o que em algumas situações exige maior esforço computacional. Tais representações denominam-se pilha e fila estática. Quando as estruturas pilhas e filas são representadas por elementos que, além de conter o dado, possuem também um ponteiro para o próximo elemento, ou seja, elementos encadeados, têm-se representações denominadas pilha e fila dinâmicas.

1.1 PILHAS

A estrutura denominada pilha é considerada do tipo FILO (*first in last out*), ou seja, o primeiro elemento inserido será o último a ser removido. Nessa estrutura, cada elemento armazena um ou vários dados (estrutura homogênea ou heterogênea, respectivamente) e um ponteiro para o próximo elemento, permitindo o encadeamento e mantendo a estrutura linear.

1.2 OPERAÇÕES COM PILHAS

Nas pilhas serão abordadas as seguintes operações: inserir na pilha, imprimir toda a pilha, remover e esvaziar. Qualquer estrutura desse tipo possui um ponteiro denominado TOPO, no qual todas as operações de inserção e remoção acontecem. Assim, as operações ocorrem sempre na mesma extremidade da estrutura.

Veja a figura a seguir, com endereços de memória meramente ilustrativos.

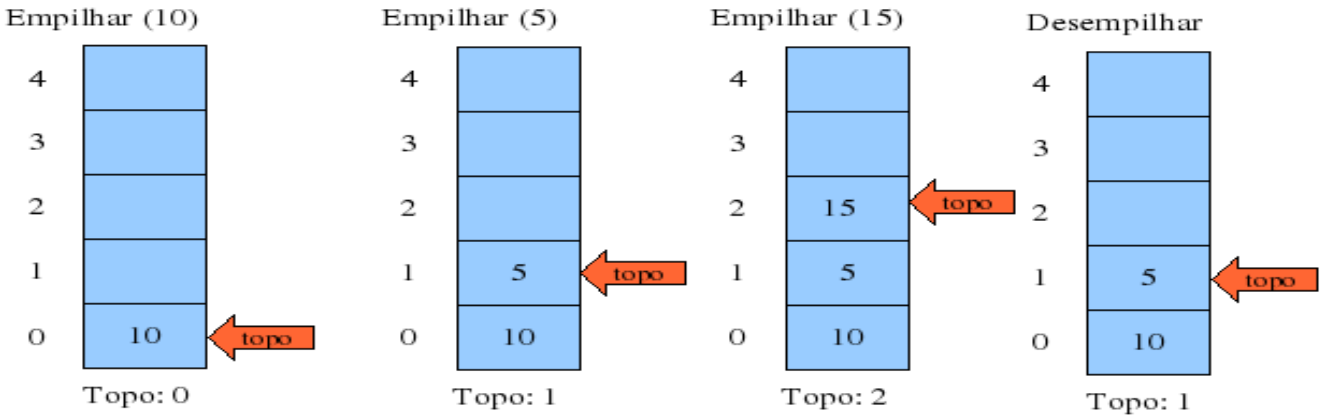

Figura 1 - Exemplo de pilha.

TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

1.3 ANÁLISE DA COMPLEXIDADE

A operação de inserção e remoção na pilha sempre realiza operações básicas, como a de atribuição, para atualizar o topo da pilha. Logo, são operações de tempo constante e gastam tempo *O(1).*

Já a operação de consultar toda a pilha percorre todos os elementos armazenados nela. Considerando que uma pilha contém *n* elementos, o tempo de execução será *O(n).*

A operação de esvaziamento da pilha consiste em remover todos os elementos armazenados nela. O tempo gasto nessa operação depende da linguagem de programação que está sendo utilizada.

- Na linguagem C/C++ é necessário desalocar cada um dos elementos da pilha, gastando tempo proporcional ao tamanho dela, ou seja *O(n)*.
- Já no caso de JAVA, não é necessário remover cada um dos elementos, apenas atualiza-se o ponteiro topo da pilha para nulo e as memórias alocadas serão desalocadas por um procedimento JAVA.

1.3 DESCRIÇÃO DO PROBLEMA

Peirópolis é um distrito rural de Uberaba, localizado às margens da rodovia BR-262, a cerca de 20 km do centro da cidade. No começo do Séc. XX, destacou-se como produtor de calcário e atualmente é uma atração turística do Município em função dos fósseis encontrados nas imediações.

O paleontólogo gaúcho Llewellyn Ivor Price(1905-1980), considerado o pai da paleontologia brasileira, começou a trabalhar em Peirópolis em 1947 e permaneceu na região até 1974. Realizou uma escavação sistemática na região de Caieira, entre 1949 e 1961. Como resultado, foram recuperadas centenas de ossos fossilizados do período Cretáceo Superior (100 a 65 milhões de anos atrás), sobretudo de dinossauros do grupo dos titanossauros.

Peirópolis encabeça o macroprojeto Geopark de Uberaba – Terra de Gigantes, voltado para desenvolver o turismo sustentável na região, e que visa obter a chancela oficial junto à Unesco a partir de 2019. A Prefeitura de Uberaba, em parceria com a UFTM, Sebrae, ABCZ, Fiemg, Comtur e demais entidades parceiras e instituições de ensino trabalham para cumprir todas as metas para consolidar a certificação.

O Geossítio Peirópolis agora conta com melhorias na acessibilidade do local, na sinalização turística, na inclusão da comunidade e dos alunos, e no monitoramento do fluxo de visitação para comprovação dos resultados.

Diante dessa parceria algumas reformas serão realizadas e locais especiais foram construídos para alguns fósseis. Sendo assim eles deverão ser removidos para esses lugares propriamente ditos.

TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

É óbvio que os fósseis são pesados e não pode ser movidos como um simples bloco. Então, eles serão cuidadosamente desmontados em partes horizontais (blocos). Um guindaste será usado para mover um bloco por vez para uma nova localização. É claro, que o guindaste pode segurar um único bloco ao mesmo tempo, e os blocos devem ser remontados em ordem apropriada, de forma que um espaço temporário é necessário para manter os blocos fora do caminho. Infelizmente, há um espaço muito limitado, e pode-se ter ao mesmo tempo **três pilhas de blocos: a localização original, a nova localização, e um espaço temporário**. O guindaste pode pegar somente o bloco no topo de uma pilha e colocá-lo no topo de uma pilha diferente.

Os especialistas mediram e determinaram o peso e a capacidade de cada bloco. A capacidade de um bloco é o máximo peso que um bloco pode suportar sobre ele. Obviamente, cada bloco é forte o suficiente para aguentar os blocos que estavam originalmente sobre ele. Caso contrário os fósseis teriam desmoronado há muito tempo atrás.

2 ESPECIFICAÇÃO DA TAD

A especificação de um tipo abstrato de dados (TAD) descreve todas as funcionalidades que devem ser implementadas para representar o tipo. Desse modo, uma TAD pode ser considerada como uma forma de documentação alto nível para uma certa estrutura de dados (ED). O TAD descreve:

(1) as informações que a ED deve prover e

(2) as operações que podem ser realizadas sobre a ED e que devem ser disponibilizadas na sua implementação.

2.1 DINOSSAURO

Um dinossauro D é definidos por um conjunto de blocos, um ponteiro TOPO que armazena a posição do último bloco que foi inserido e uma variável que armazena a quantidade de blocos que esse dinossauro contém.

A implementação da ED que representará um dinossauro deverá fornecer o tipo Dinossauro como ponteiro para struct (em C). As operações realizadas na ED esclarecerão quais informações devem ser providas na implementação.

Exemplo:

```
struct dinossauro { . . . } ;
typedef struct dinossauro *Dinossauro;
```

TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

2.2 BLOCO

Os blocos são cada parte que compõe os fósseis, sendo assim deverão armazenar o seu id(identificador), peso e a capacidade (peso que ele suporta sobre ele).

Cada dinossauro é composto por vários blocos empilhados segundo uma ordem específica.

2.3 OPERAÇÕES

Para a implementação das rotinas que executam as operações são adotadas as seguintes convenções:

- O nome das rotinas é padronizado: tem sempre um prefixo de duas letras maiúsculas seguindo de um sufixo minúsculo. A primeira letra do prefixo é D e a segunda indica o tipo de retorno da rotina, caso seja: ponteiro Dinossauro (D), ponteiro Bloco (B), número inteiro (I), número real (F), valor booleano(BO) e assim por diante. O sufixo especifica a tarefa realizada na operação.
- Todas as rotinas recebem um ponteiro Dinossauro como parâmetro de entrada, que é assumido ser um ponteiro válido. Esse pré-requisito não precisa ser checado.
- Nenhuma rotina imprimirá mensagem de erro, se for constatado um erro então a rotina será interrompida e retornará um código de erro convencionado (geralmente zero ou nulo).
- Abaixo segue algumas rotinas/operações de que podem ser implementadas. A equipe deverá analisar o problema a ser resolvido e especificar as operações necessárias, bem como a descrição de cada uma.

Algumas operações a serem realizadas seguem abaixo, agrupadas por afinidade entre as funcionalidades:

Operação:: DDcriaDinossauro	
Descrição:: cria um dinossauro	
Entradas::	**Saída:**
- b:: número máximo de blocos	– p:: ponteiro para o dinossauro criado D
Pré-requisitos::	**Pós-requisitos::**
- b >= 2;	- D tem memória alocada para os blocos;

TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

Operação:: DDdestroiDinossauro	
Descrição:: destrói um dinossauro	
Entradas::	**Saída:**
- p:: ponteiro para o dinossauro D	– p:: ponteiro nulo
Pré-requisitos::	**Pós-requisitos::**
- não tem	- a memória alocada para D foi liberada;

Operação:: DBOadicionaBloco	
Descrição:: adiciona um bloco no dinossauro	
Entradas::	**Saída:**
- d: ponteiro para dinossauro - p: peso do bloco - c: capacidade do bloco	- b: booleano indicando se conseguiu adicionar o bloco
Pré-requisitos::	**Pós-requisitos::**
- quantidade de blocos tem que ser menor do que o número máximo de blocos suportado pelo dinossauro.	- TOPO é atualizado;

Operação:: DBremoveBloco	
Descrição:: remove um bloco	
Entradas::	**Saída:**
- d: ponteiro para dinossauro	- p: ponteiro para o bloco removido
Pré-requisitos::	**Pós-requisitos::**
- TOPO diferente de Null;	- TOPO é atualizado;

Operação:: DBOsalvaDinossauro	
Descrição:: salva o dinossauro em um arquivo (conforme o exemplo)	
Entradas::	**Saída:**
- d: ponteiro para dinossauro - f: nome do arquivo de saída	- bo: booleano indicado se conseguiu salvar o dinossauro.
Pré-requisitos::	**Pós-requisitos::**
- não tem	- arquivo gerado.

TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

Operação:: DDcarregaDinossauro	
Descrição:: cria e carrega um dinossauro armazenado em um arquivo	
Entradas::	**Saída:**
- f: nome do arquivo que contém o dinossauro	-p: ponteiro para o dinossauro.
Pré-requisitos::	**Pós-requisitos::**
- nome do arquivo válido	- ponteiro para o dinossauro ou nulo caso o arquivo seja inválido.

3 TAREFA

Cada grupo deverá desenvolver duas tarefas:

1. desenvolver uma estrutura de dados para representar dinossauros, isto é, implementar um módulo para prover as funcionalidades para a manipulação da TAD. Produtos gerados: os arquivos.

2. desenvolver um procedimento que, planeje como mover todos os blocos da localização original até a nova localização, em uma ordem apropriada. Seu objetivo é completar o trabalho usando a menor quantidade de movimentos possíveis (um movimento consiste em pegar um bloco do topo de uma pilha e colocá-lo no topo de uma pilha diferente). Esta é uma tarefa que deverá produzir um arquivo de saída contendo todos os movimentos realizados. **Lembrando que só temos disponíveis três pilhas (original, a nova e uma temporária). Protótipo para o procedimento:**

- **void moveDinossauro(Dinossauro d);**
- Por exemplo, dado como entrada o dinossauro da tabela 1, o procedimento deve gerar o arquivo de saída ilustrado na mesma tabela. **Podemos ter mais de uma solução no arquivo de saída, depende da implementação adotada.**

Arquivos de Entrada:: Os arquivos de entrada são formatados da seguinte maneira:

- Primeira linha: um inteiro N ($2 \leq N \leq 10$) que representa o número de blocos horizontais;
- As próximas N linhas descrevem os N blocos horizontais em ordem, do topo até a base do dinossauro. Cada descrição de bloco consiste de 3 inteiros que são o ide, o peso e a capacidade do bloco. O peso e a capacidade variam de 1 a 1000, inclusive.

Arquivos de Saída:: O arquivo de saída deve apresentar os movimentos realizados pelo guindaste. Cada movimento deve ser descrito em uma linha. Então, cada linha contém três inteiros: o id, a pilha fonte e pilha destino, separados por um simples espaço.

Observe que a pilha 1, 2 e 3 representam a localização original, o espaço temporário e a nova localização do dinossauro, respectivamente.

Tabela 1 - Exemplo de entrada e saída.

Arquivo de Entrada 1	Arquivo de Saída 1
4 4 3 4 3 2 3 2 3 6 1 2 10	4 1 3 3 1 3 2 1 2 3 3 2 4 3 2 1 1 3 4 2 1 3 2 1 2 2 3 3 1 3 4 1 3

4 EXEMPLOS DE CÓDIGO

Nesta seção serão apresentados como exemplos os códigos em C que implementam tarefas comuns realizadas sobre pilhas, utilizando a implementação da TAD proposta. É interessante notar que a implementação da TAD armazena apenas informações sobre a estrutura do dinossauro.

4.1 CRIAR E SALVAR UM DINOSSAURO

```c
#include <stdio.h>
#include <stdlib.h>
#include "dinossauro.h"

int main()
{
    int i;

    Dinossauro d = DDcriaDinossauro(6);

    //cria 6 blocos
    DBcriaBloco(d, 10, 200);
    DBcriaBloco(d, 20, 100);
    DBcriaBloco(d, 15, 100);
    DBcriaBloco(d, 25, 50);
    DBcriaBloco(d, 25, 50);
    DBcriaBloco(d, 5, 2);

    //salva dinossauro
    DBsalvaDinossauro(d,"dinossauro.txt") ;
    DBdestroiDinossauro(d) ;

    return 0;
}
```

TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

4.2 CARREGAR UM DINOSSAURO SALVO EM UM ARQUIVO E IMPRIMIR SEUS BLOCOS

```c
#include <stdio.h>
#include <stdlib.h>
#include "dinossauro.h"

int main()
{
    int i;
    Bloco b;

    Dinossauro d = DDCarregaDinossauro("dinossauro.txt");

    //imprime todos os blocos
    for (i = 1; i<=dinossauro.quantidadeBlocos; i++){
        b =dinossauro.blocos[i];
        printf("%d %d", b.peso, b.capacidade);
    }

    DBdestroiDinossauro(d);
    return 0;
}
```

5 CRITÉRIOS DE CORREÇÃO

Serão adotados os seguintes critérios de correção para o trabalho:

1. Correção: somente serão corrigidos códigos portáveis e sem de erros de compilação;

2. Precisão: execução correta numa bateria de testes práticos; (70%)

3. Modularização: uso adequado da TAD e das estruturas de dados; (30%)

4. Qualidade do código fonte: legibilidade, indentação, uso adequado de comentários; (requisito)

5. Documentação: relatório, conforme instruções apresentadas a seguir. (requisito)

O critério "correção" é obrigatório e condiciona a avaliação toda, o seu código deve compilar senão seu trabalho não será aceito e vai receber nota zero. Alguns critérios não receberão pontuação específica, mas funcionarão como "requisitos" que poderão afetar

a pontuação se não forem atendidos adequadamente, portanto leve-os em conta durante o desenvolvimento do trabalho.

Cada grupo deverá desenvolver um módulo de implementação para a TAD e um módulo para o procedimento de mover os blocos.

O programa principal que vai testar esses módulos, nos dois casos, será fornecido pelo professor da disciplina. O programa principal vai ler os arquivos de entrada pré-determinados, vai fazer uso do módulo Dinossauro desenvolvido pelo grupo, e vai gerar as saídas correspondentes, que deverão ser idênticas àquelas associadas aos arquivos de entrada pré-determinados. Se o módulo não funcionar exatamente como foi especificado na TAD, o teste vai gerar erro e impactar na pontuação do trabalho.

Haverá uma apresentação oral e individual do trabalho. Todos os integrantes do grupo devem participar dessa apresentação.

Na ausência de plágio, as notas dos trabalhos corretos serão computadas individualmente da seguinte forma: nota = nota_apresentacao $*$ nota_trabalho, ou seja, a nota final é ponderada pela nota da apresentação.

5.1 DOCUMENTAÇÃO

Esta seção descreve o formato e o conteúdo do relatório que deve ser gerado como produto final do trabalho. Seja sucinto: em geral é possível escrever um bom relatório entre 2 e 4 páginas.

O relatório documentando seu sistema deve conter as seguintes informações:

1. introdução: descrever o problema resolvido e apresentar uma visão geral do sistema implementado;

2. implementação: descrever as decisões de projeto e implementação do programa. Essa parte da documentação deve mostrar como as estruturas de dados foram planejadas e implementadas. Sugestão: mostre uma figura ilustrativa da TAD, os tipos definidos, detalhes de implementação e especificação que porventura estavam omissos no enunciado, etc.

3. validação: descrição dos testes que o grupo fez para validar o trabalho, se seguiu alguma metodologia etc.

4. conclusão: avaliação do grupo sobre o trabalho considerando a experiência adquirida, a contribuição para o aprendizado da disciplina, as principais dificuldades encontradas ao implementá-lo e como tais dificuldades foram superadas;

5. bibliografia: cite as fontes consultadas na resolução do trabalho, se houver

TRABALHO PRÁTICO – 20,0 PONTOS

ENTREGA: 26/03/2021

DISCIPLINA: ESTRUTURAS DE DADOS I

PROFA. PATRÍCIA PROENÇA

REFERÊNCIAS

[1] ASCENCIO, A. F. , Araújo, G. S. Estruturas de Dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Prentice Hall, 2010.

[2] ZIVIANI, N. Projeto de algoritmos: com implementações em pascal e c, 2a ed. rev. e ampl. São Paulo: Pioneira Thomson Learning, 2005. 552 p.

[3] http://www.uftm.edu.br/museudosdinossauros/index.php/fosseis