UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO – CAMPUS PAU DOS FERROS
BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO
DISCIPLINA: LABORATÓRIO DE ALGORITMOS E ESTRUTURAS DE DADOS I

LISTA DE EXERCÍCIOS
TIPOS ABSTRATOS DE DADOS

Questão 1) Crie um TAD chamado ContaBancaria que possui os seguintes campos: titular, número e saldo. Como operações, considere as seguintes:

- Cria conta: aloca dinamicamente uma estrutura do tipo ContaBancaria e retorna seu endereço. Os campos titular, número e saldo devem ser fornecidos como parâmetros;
- Deposita: recebe, como parâmetros, o endereço de uma estrutura do tipo ContaBancaria e um valor, atualizando o saldo;
- Saca: recebe, como parâmetros, o endereço de uma estrutura do tipo ContaBancaria e um valor, atualizando o saldo. Verificar se o saldo é suficiente para realizar o saque;
- Transfere: recebe, como parâmetros, os endereços das estruturas do tipo ContaBancaria e um valor, atualizando os saldos. Verificar se o saldo é suficiente para realizar a transferência;
- Saldo: recebe o endereço de uma estrutura do tipo ContaBancaria e retorna seu saldo;
- Exclui conta: libera o espaço alocado dinamicamente para a estrutura.

Faça o que se pede nos itens a seguir:

a) Crie um arquivo (*contabancaria.h*) com a interface do TAD.
b) Crie um arquivo (*contabancaria.c*) com a implementação do TAD.
c) Crie um programa que utiliza o TAD ContaBancaria.

Questão 2) Crie um tipo abstrato de dados para representar um aluno. A interface do TAD é representada pelo arquivo *aluno.h* a seguir:

```
/* Arquivo aluno.h */

#include <stdio.h>

typedef struct aluno Aluno;

Aluno* aluno_cria(char* nome, float nota);

void aluno_libera(Aluno* a);

void aluno_imprime(Aluno* a);

void aluno_ordena(int n, Aluno** v);

void aluno_salva(FILE* fp, int n, Aluno** v);
```

Escreva o arquivo *aluno.c* com a implementação do tipo e das funções exportadas pelo TAD. Leve em consideração o seguinte:

- Um aluno possui um nome e uma nota;
- A função aluno_cria aloca dinamicamente uma estrutura Aluno, configurando seu nome e nota com valores passados como parâmetros;
- A função aluno_libera libera o espaço em memória para a estrutura do tipo Aluno;
- A função aluno_imprime imprime na saída padrão os dados de um aluno;
- A função aluno_ordena ordena em ordem alfabética um vetor de ponteiros para estruturas do tipo Aluno;
- A função aluno_salva salva em um arquivo texto os dados de um vetor de ponteiros para estruturas do tipo Aluno; use o seguinte formato para cada linha do arquivo: nome_do_aluno\tnota_do_aluno;

Após implementar o TAD, escreva uma função main para teste, realizando as seguintes tarefas nessa ordem:

- Crie cinco alunos, armazenando-os em um vetor de ponteiros com cinco elementos;
- Imprima os dados dos cinco alunos criados;
- Ordene o vetor de ponteiros;

- Salve os dados dos alunos em um arquivo texto;
- Libere o espaço em memória alocado para os cinco alunos, não esquecendo de atualizar os valores dos ponteiros adequadamente;

Questão 3*) Considere a criação de um tipo abstrato de dados para representar matrizes de valores reais alocadas dinamicamente, com dimensões m por n fornecidas em tempo de execução. O TAD Matriz (denominação do tipo) deve fornecer as seguintes operações:

- Cria: operação que cria uma matriz de dimensão m por n;
- Libera: operação que libera a memória alocada para a matriz;
- Acessa: operação que acessa o elemento da linha i e da coluna j da matriz;
- Atribui: operação que atribui o elemento da linha i e da coluna j da matriz;
- Linhas: operação que retorna o número de linhas da matriz;
- Colunas: operação que retorna o número de colunas da matriz.

A interface do módulo (arquivo *matriz.h*) pode ser dada pelo código a seguir:

```
typedef struct matriz Matriz;

Matriz* mat_cria(int m, int n);
void mat_libera(Matriz* mat);
float mat_acessa(Matriz* mat, int i, int j);
void mat_atribui(Matriz* mat, int i, int j, float v);
int mat_linhas(Matriz* mat);
int mat_colunas(Matriz* mat);
```

Considere agora as estruturas a seguir que definem matrizes dinâmicas representadas por vetor simples e por vetor de ponteiros, respectivamente:

```
struct matriz {
        int lin;
        int col;
        float* v;
};
```

```
struct matriz {
        int lin;
        int col;
        float** v;
};
```

Dadas essas considerações, sua tarefa é implementar as funções do TAD a partir das duas estratégias alternativas para representação de matrizes dinâmicas. Perceba que, independente da estratégia utilizada, a funcionalidade oferecida pelo tipo abstrato não se altera.

*Fonte:

CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. **Introdução a Estruturas de Dados: com técnicas de programação em C**. Elsevier, 2016