LISTA1 - N2 - ESTRUTURA DE DADOS

Vetores, ordenação, recursividade, ponteiros, registros e estruturas de dados dinâmicas.

> Prof. Ed

Instruções:

- Os programas **NÃO DEVEM SER COMPACTADOS**. O código-fonte deve ser enviado via upload diretamente na resposta do exercício (arquivo por aquivo)
- Cada arquivo deve ter o seguinte formato: `ED-lista2-questaoXX` onde `XX` é o número da questão correspondente.
- O trabalho é em **DUPLAS** (porém CADA ALUNO DEVE ENVIAR OS ARQUIVOS INDIVIDUALMENTE)
- **IMPORTANTE**: NÃO SERÃO ACEITOS TRABALHOS QUE NÃO ESTIVEREM NO FORMATO ACIMA
- OBSERVAÇÃO: TODOS os programas entregues devem ter o seguinte cabeçalho:

```
/*
**    Função :
**    Autor :
**    Data  :
**    Observações:
*/
```

  Onde deverá estar escrito o que o programa faz, o autor (nome, turma, a data e as observações que forem pertinentes. Os trabalhos **não serão aceitos** após a data SOB HIPÓTESE ALGUMA.

---

1. Implemente uma função que classifica os elementos de um vetor em ordem crescente usando o seguinte algoritmo (conhecido como **"classificação por seleção"**):
   - procure pelo **menor** elemento no vetor e **permute** esse elemento com o **primeiro elemento** do vetor;

- repita este processo para o **subvetor** que se inicia no segundo elemento e, assim, sucessivamente;
- o processo termina quando o **subvetor** contiver apenas um elemento.
- Teste a função com dados gerados **aleatoriamente**.

2. Defina um **registro** (**estrutura - struct**) empregado para armazenar os dados (nome, data de nascimento, RG, data de admissão e salário)de um **empregado** de uma empresa. Criar um novo tipo de dados chamado **Empregado** usando a estrutura empregado. **Defina um vetor** de **empregados** (usando **alocação dinâmica**) para armazenar todos os empregados de sua empresa. Implementar **rotinas para ler, escrever e excluir** registros deste tipo.

3. Suponha que uma **empresa aérea** mantém um cadastro de aeroportos como um **vetor de ponteiros** para estruturas que contêm as seguintes informações:
   a. **Sigla**: **string** com até 3 caracteres;
   b.**Cidade**: **string** com até 50 caracteres;
   c.**Pais**: **string** com até 30caracteres;
   d.**Taxa**: um valor **real**;
   e.**Capacidade**: um valor **inteiro**;
   Defina uma estrutura em C com o nome de **aeroporto**, que tenha os campos apropriados para guardar todas as informações descritas anteriormente. Defina também **um novo tipo** de dados com o nome de **Aeroporto**, correspondendo a essa estrutura. **Defina um vetor de Aeroportos** (usando alocação dinâmica para a quantidade de aeroportos) para armazenar todos os aeroportos que a empresa aérea trabalha. Implementar **rotinas para ler, escrever e excluir** registros deste tipo.

4. A famosa **Conjectura de Goldbach** diz que **todo inteiro par maior que 2** é a soma de dois outros números primos. Testes extensivos foram feitos sem, contudo, ser encontrado um contra-exemplo. **Escreva um programa que mostre que a afirmação é verdadeira** para todo número par entre **700** e **1100**. O programa deve imprimir cada número e os primos correspondentes.

5. Refaça o programa anterior (com outro nome) para que o computador teste de **2** até um número **N** informado pelo usuário

6. O **método de Newton-Raphson** é utilizado para calcular **aproximadamente raízes de funções (ou de equações).** Ele se baseia na seguinte premissa:

- "**Se** $f(x) = 0$ tem apenas uma raiz no intervalo $[a, b]$ e se **nem** $f'(x)$ nem $f''(x)$ se anulam nesse intervalo, **escolhido** x_0 como aquele dos dois números a e b para o qual $f'(x)$ e $f''(x)$ tem mesmo sinal, **então** $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ situa-se mais perto da raiz do

que x_0 onde $f'(x_k)$ é a derivada da **função** $f(x)$**.** O Método de Newton-Raphson é excelente para calcular aproximações de raízes reais de funções reais, convergindo rapidamente.

Sabendo disso, as calculadoras científicas usam este método para **calcular a raiz quadrada de números n**, usando a seguinte dedução: $f(x) = x^2 - n$ e $f'(x) = 2x$; portanto para calcular $\sqrt{n}$ basta fazer $f(x) = x^2 - n$ **e** $f'(x) = 2x$ para qualquer n e executar o algoritmo iterativo. Pesquise em livros ou na Internet como funciona o método de Newton, crie um algoritmo e aplique-o em uma função para calcular **a raiz quadrada de um número n com aproximação de 0.0001**.

7. Implemente uma função que classifica os elementos de um vetor em ordem crescente usando o algoritmo "**quicksort**", que pode ser estabelecido da forma a seguir:

    1. Seja `m` o elemento do vetor que ocupa a posição "central" no vetor;
    2. Seja `i` o índice do primeiro e `j` o índice do último elemento do vetor;
    3. Enquanto `i` for menor ou igual a `j`, faça com que:
        - O valor de `i` aumente até encontrar um elemento maior do que `m`;
        - O valor de `j` diminua até encontrar um elemento menor que `m`;
        - Haja troca entre os elementos que ocupam as posições `i` e `j`;
    4. Ao final desses passos a situação do vetor será a seguinte: à esquerda da posição central, existem somente elementos menores que `m` e à direita da posição central, existem somente elementos maiores que `m`.

        Assim, o problema de ordenar o vetor se reduz a um problema de ordenar cada uma dessas "metades". Para ordenar, então, basta aplicar os mesmos passos a cada uma das "metades" **RECURSIVAMENTE**.

    5. Teste a função com valores gerados **aleatoriamente**

8. Crie um novo TAD (Tipo Abstrato de Dado) de arquivo único chamado **Complexo** para realizar aritmética com números complexos. Utilize variáveis **double** para representar os campos deste tipo. Implemente funções para as seguintes operações: **criar** e **destruir** um número complexo; **ler** um número complexo, **somar** dois números complexos, **subtrair** dois números complexos, **multiplicar** dois números complexos, **dividir** dois números complexos, **mostrar** um número complexo na forma (a,b) onde a é a **parte real** e b, a **parte imaginária.**

9. Crie uma função para ordenar elementos de um vetor (pode ser de inteiros ou reais) usando o "método BubbleSort" (ordenação Bolha). Em seguida realize a CONTAGEM do

números de `if's` realizados e a quantidade de trocas realizadas. Mostre, ao final, após o vetor estar ordenado, a quantidade de `if's` e de `trocas` realizadas. Use a função em um programa que solicita a quantidade de elementos do vetor para o usuário e preenche este vetor com valores aleatórios.