Exercícios aula 20

Faça os exercícios requeridos

Exercício 01 (individual)

- Utilizando os conceitos apresentados na aula 19, implemente um programa com uma estrutura de árvore de busca para a seguinte estrutura de dados:
  - ID
    - Codigo (longInt)
    - nome (String)
    - filiacao(String)
    - dataNasc (tipo Data – criar a TAD)
    - CidadeNatal (String)

Exer 01

- Faça um menu com todas as possíveis opções ao usuário:
  - Inserir
  - Remover
  - Buscar
  - Mostrar todos os Registros cadastrados

Exercício 2

- Implemente uma função recursiva de um algoritmo para busca binária. Coloque tal algoritmo em um programa que permita a leitura de dados em um vetor e a realização da busca por meio do algoritmo recursivo.

Exercício 3

- O Método BubbleSort (Troca) consiste de uma estratégia de ordenação na qual comparam-se dois elementos e trocam-se suas posições se o segundo elemento é menor do que o primeiro.

  São feitas várias passagens pela tabela. Em cada passagem, comparam-se dois elementos adjacentes. Se estes elementos estiverem fora de ordem, eles são trocados
- Implemente um programa que leia dados do tipo ID (Exer 1) em um vetor (tamanho máximo de 100) e depois utilize o método bubble sort para ordenar os registros no vetor.

Exercício 4

- Há muitas maneiras de implementar a busca binária. Os exercícios abaixo discutem algumas implementações. Todas as funções procuram encontrar x em v[0..n-1]. Todas as funções produzem (ou deveriam produzir) j em 0..n tal que $v[j-1] < x \leq v[j]$.
- A implementação da busca binária exige cuidado e atenção aos detalhes; é muito fácil escrever um algoritmo que dá respostas erradas ou entra em loop.

Exer 4.1

- Mostre que a seguinte alternativa de buscabinaria2 funciona corretamente. Ela é quase tão bonita quanto a versão discutida acima.

```
e = 0; d = n;
   while (e < d) {          // v[e-1] < x <= v[d]
      m = (e + d)/2;
      if (v[m] < x) e = m+1;
      else d = m;
   }                        // e == d
   return d;
```

- Que acontece se trocarmos while (e < d) por while (e <= d)? Que acontece se trocarmos (e+d)/2 por (e-1+d)/2?

Exer 4.2

- Mostre que a seguinte alternativa de buscabinaria2 funciona corretamente. Eu acho que ela é um pouco mais feia que as versões anteriores.

```
e = 0; d = n-1;
  while (e <= d) {        // v[e-1] < x <= v[d+1]
    m = (e + d)/2;
    if (v[m] < x) e = m+1;
    else d = m-1;
  }                       // e == d+1
  return d+1;
```

Exer 4.3

- A seguinte alternativa de buscabinaria2 funciona corretamente?

```
e = -1; d = n-1;
  while (e < d) {
     m = (e + d)/2;
     if (v[m] < x) e = m;
     else d = m-1;
  }
  return d+1;
```

Exer 4.4

- A seguinte alternativa de buscabinaria2 funciona corretamente?

```
e = -1; d = n-1;
  while (e < d) {
     m = (e + d + 1)/2;
     if (v[m] < x) e = m;
     else d = m-1;
  }
  return d+1;
```

Exer 4.5

- Preencha os ?? corretamente.

```
int buscabinaria2( int x, int n, int v[]) {
    int e = ??,  d = ??;
    while (e ?? d-1) {
        m = (e + d)/2;
        if (v[m] ?? x) e = m;
        else d = m; }
    return ??; }
```