Estácio

Voltar

Teste de Conhecimento

avalie sua aprendizagem

ESTRUTURA DE DADOS
CCT0826_AS_202101110137_V1

Lupa Calc.

Aluno: **DOUGLAS MATOS DA SILVA** Matr.: **202101110137**
Disc.: **ESTRUTURA DE DADOS** **2022.1 EAD (G) / EX**

Prezado (a) Aluno(a),

Você fará agora seu **TESTE DE CONHECIMENTO**! Lembre-se que este exercício é opcional, mas não valerá ponto para sua avaliação. O mesmo será composto de questões de múltipla escolha.

Após responde cada questão, você terá acesso ao gabarito comentado e/ou à explicação da mesma. Aproveite para se familiarizar com este modelo de questões que será usado na sua AV e AVS.

1. São métodos ou algoritmos conhecidos de ordenação de dados por troca:

- ( ) busca por ordenação e ordenação shell.
- (✔) bubble sort e quicksort.
- ( ) quicksort e hashing.
- ( ) ordenação shell e hashing.
- ( ) hashing e bubble sort.

Explicação:

Bubble sort é o algoritmo mais simples, mas o menos eficientes. Neste algoritmo cada elemento da posição i será comparado com o elemento da posição i + 1, ou seja, um elemento da posição 2 será comparado com o elemento da posição 3. Caso o elemento da posição 2 for maior que o da posição 3, eles trocam de lugar e assim sucessivamente. Por causa dessa forma de execução, o vetor terá que ser percorrido quantas vezes que for necessária, tornando o algoritmo ineficiente para listas muito grandes.

O Quicksort é o algoritmo mais eficiente na ordenação por comparação. Nele se escolhe um elemento chamado de pivô, a partir disto é organizada a lista para que todos os números anteriores a ele sejam menores que ele, e todos os números posteriores a ele sejam maiores que ele. Ao final desse processo o número pivô já está em sua posição final. Os dois grupos desordenados recursivamente sofreram o mesmo processo até que a lista esteja ordenada.

2. Considere as seguintes afirmações: I. Só podemos ter uma matriz de no máximo duas dimensões. Exemplo: C[100][100]. II. Ao declararmos um vetor int A[10], se escrevemos A[2] acessamos o segundo elemento do vetor. III. Uma string declarada como char B[30] armazena no máximo 30 caracteres. Escolha a alternativa correta:

- ( ) Estão corretas apenas as afirmativas I e III.
- (✔) Nenhuma afirmação está correta.
- ( ) Está correta apenas a afirmativa II.
- ( ) Estão corretas apenas as afirmativas I e II.
- ( ) Está correta apenas a afirmativa I.

Explicação:

Analisando cada afirmativa :

I. Só podemos ter uma matriz de no máximo duas dimensões. Exemplo: C[100][100].

Falso. Podemos ter matrizes unidimensionais, tridimensionais, etc...

II. Ao declararmos um vetor int A[10], se escrevemos A[2] acessamos o segundo elemento do vetor.

Falso. Em C++ o índice inicial é zero. Logo, A[2] é o elemento de índice 2, ou seja, o 3o. elemento do vetor.

III. Uma string declarada como char B[30] armazena no máximo 30 caracteres. Escolha a alternativa correta:

Falso. B pode armazenar no máximo 29 caracteres que sejam dados, pois existe uma área para o caracter nulo.

Logo, todas as opções são falsas.

3. Estude atentamente o código a segir:

```
int deciframe(int v[ ], int tam, int e){
int i = 0, f = tam -1, m;
while ( i <= f ){
m = ( i + f ) / 2;
if ( v[m] == e ) { return m; }
if ( e < v[m] ) { f = m - 1; }
else { i = m + 1; }
}
return -1;
}
```

Sabendo que a chamada da mesma foi feita com os parâmetros recebendo os seguintes valores, o que ela retornaria?

```
v[10] = {0, 2, 4, 6, 8, 10, 20, 100}
tam = 8
e = -6
```

- ( ) 4
- (✔) -1
- ( ) 3
- ( ) 0
- ( ) 6

Explicação:

Analisando

```
int deciframe(int v[ ], int tam, int e)  {
int i = 0, f = tam -1, m;
while ( i <= f ){
  m = ( i + f ) / 2;
  if ( v[m] == e ) { return m; }
  if ( e < v[m] )
     { f = m - 1; }
  else
     { i = m + 1; }
}
return -1;
}
```

Sabendo que a chamada da mesma foi feita com os parâmetros recebendo os seguintes valores, o que ela retornaria?

```
v[10] = {0, 2, 4, 6, 8, 10, 20, 100}
tam = 8
e = -6
```

Está procurando pelo valor e = -6. Como se sabe pelo estudo da busca binária e olhando o código dado, temos que -6 não existe na lista então a função irá retornar -1 (vide última linha da função). SE a busca tivesse sucesso, o teste do 1o. if no while seria verdadeiro e então o índice m de v seria retornado. Mas não é o caso, pois -6 não foi encontrado.

4. As estruturas de dados são utilizadas para manter dados ou informações organizados na memória, o que possibilita a otimização do uso destes dados. Porém, as estruturas guardam características especiais na manipulação destes dados, assim deve-se escolher a estrutura certa ou mais adequada para uma determinada aplicação. Portanto marque a opção que representa a melhor estrutura, quando se tem como requisitos principais o acesso aleatório aos dados e alocação destes de forma contínua na memória.

- (✔) Lista Sequencial
- ( ) Pilha Sequencial
- ( ) Fila Sequencial
- ( ) Pilha Encadeada
- ( ) Lista Encadeada

Gabarito Comentado

5. Considere uma lista sequencial L com n fichas de professores, sendo que cada ficha de professor é modelada pela struct :

```
struct professor {
int matricula;
char titulo[30];
};
```

e a lista L é assim declarada : professor L[n];

Assinale o trecho que corretamente exibe todas as matrículas e titulações de todos os n professores de L .

- for (int i = 0; i < n; i++) cout << L[i].matricula << " " << L[i].titulo[30] << endl;
- ✔ for (int i = 0; i < n; i++) cout << L[i].matricula << " " << L[i].titulo << endl;
- for (int i = 0; i < n; i++) cout << L.matricula[i] << " " << L.titulo[i] << endl;
- for (int i = 0; i < n; i++) cout << L[i] << endl;
- for (int i = 0; i < n; i++) cout << L.matricula << " " << L.titulo << endl;

Explicação:

Como L é um vetor de n elementos do tipo professor. Então, para percorrer o vetor de índice i temos que fazer L[i] seguido do ponto seguido do campo, que pode ser matricula ou titulo.

Assiim, a opção correta é

```
for (int i = 0; i < n; i++)
   cout << L[i].matricula << " " << L[i].titulo << endl;
```

6. _____ é uma lista linear em que a alocação de memória pode ser estática, e que a forma de armazenamento é contígua ou sequencial na memória. Usamos este tipo de lista quando se tem em mente um tamanho pré-definido, ou seja, quando se sabe até onde a lista pode crescer.

- Lista Linear de Alocação de Memória
- Lista Linear Não Alocada
- Lista Não Linear
- Lista Linear Não Sequencial
- ✔ Lista Linear Sequencial

7. Considere uma lista sequencial **L** com **n** fichas de alunos, onde cada ficha de aluno é caracterizada pela **struct** :

```
struct aluno {

  int matricula;

  char nome[30];

};
```

e a lista L é assim declarada : **aluno L[n];**

Assinale o trecho que corretamente exibe todas as matrículas e nomes de todos os **n** alunos de **L** .

- for (int i = 0; i < n; i++) cout << L.matricula << " " << L.nome << endl;
- for (int i = 0; i < n; i++) cout << L[i].matricula << " " << L[i].nome[30] << endl;
- ✔ for (int i = 0; i < n; i++) cout << L[i].matricula << " " << L[i].nome << endl;
- for (int i = 0; i < n; i++) cout << L.matricula[i] << " " << L.nome[i] << endl;
- for (int i = 0; i < n; i++) cout << L[i] << endl;

Gabarito Comentado

8. Quanto a Pesquisa ou Busca Binária julgue os itens em V (VERDADEIRO) ou F (FALSO):

- ✔ [F] A Busca Binária é mais eficiente quando o vetor não está ordenado.
- ✔ [F] Na operação de inserção de um valor do vetor passado como parâmetro, não é necessário primeiro verificar se a lista está cheia.
- ✔ [V] A Pesquisa Binária consiste em fazer uma busca em um vetor ordenado, dividindo o espaço de busca ao meio e verificando se o dado está no meio ou antes do meio ou depois do meio, comparando o valor de busca com o elemento da lista.
- ✔ [F] Na operação de remoção de um valor do vetor passado como parâmetro, não é necessário primeiro verificar se a lista está vazia.
- ✔ [V] O protótipo abaixo é válido para uma função de busca binária: int buscaBinaria(float v[], float valor , int n);

Explicação:

>> A Pesquisa Binária consiste em fazer uma busca em um vetor ordenado, dividindo o espaço de busca ao meio e verificando se o dado está no meio ou antes do meio ou depois do meio, comparando o valor de busca com o elemento da lista.

Resposta: Correto. Descreve exatamente as características.

>> O protótipo abaixo é válido para uma função de busca binária: int buscaBinaria(float v[], float valor , int n);

Resposta : Correto. É preciso termos o vetor com o conjunto de dados, o valor a ser procurado e o número de elementos.

>> A Busca Binária é mais eficiente quando o vetor não está ordenado.

Resposta: FAlso. O vetor não pode estar ordenado na busca binária.

>> Na operação de remoção de um valor do vetor passado como parâmetro, não é necessário primeiro verificar se a lista está vazia.

Resposta : É preciso verificar se a lista está vazia antes de realizar a remoção, pois não se pode retirar valor alguma de lista vazia.

>> Na operação de inserção de um valor do vetor passado como parâmetro, não é necessário primeiro verificar se a lista está cheia.

Resposta : É preciso verificar se a lista está cheia antes de realizar a inserção. pois não se pode inserir valor em vetor sem espaço..

Exercício iniciado em 17/04/2022 19:24:42.