Classificação e Pesquisa

Prof. Esp. Pedro Luís Antonelli
Anhanguera Educacional

Plano de Ensino e Aprendizagem (PEA)

Anhanguera
Aqui o seu esforço ganha força.

Cronograma de Aulas	
Semana n°.	**Tema**
1	Apresentação da Disciplina e Metodologia de Trabalho. Introdução à Classificação e Pesquisa.
2	Pesquisa de Dados: método sequencial. Conceito, exemplos e exercícios.
3	Pesquisa Binária. Conceito, funcionamento, exemplos e exercícios
4	Métodos de ordenação: seleção, troca.
5	Métodos de ordenação: distribuição.
6	Métodos de ordenação: inserção.
7	Métodos de ordenação: intercalação.
8	Revisão e exercícios.
9	Atividades de Avaliação.
10	Árvores de pesquisa, árvores binárias de pesquisa.
11	Árvores de pesquisa, árvores binárias de pesquisa.
12	Árvores AVL.
13	Árvores AVL.
14	Árvores Patrícia.
15	B-Trees.
16	Tabelas hash - estática.
17	Tabelas hash - dinâmica.
18	Prova Escrita Oficial
19	Revisão
20	Prova Substitutiva

Pesquisa (ou Busca) de Dados

A Pesquisa (ou Busca) de Dados é a parte da ciência do computação dedicada ao estudo de como **recuperar informação** a partir de uma massa grande de informação previamente armazenada.

A informação é dividida em **registros,** onde cada registro possui uma **chave** para ser usada na pesquisa.

0 objetivo da pesquisa é encontrar uma ou mais ocorrências de registros com chaves iguais à **chave de pesquisa.** Quando encontrada, dizemos que ocorreu uma **pesquisa com sucesso;** caso contrário a pesquisa é **sem sucesso.**

Pesquisa (ou Busca) de Dados

Um conjunto de registros é chamado de **tabela** ou **arquivo**.

Geralmente o termo **tabela** é associado a entidades de **vida curta**, criadas na **memória interna** durante a execução de um programa.

Já o termo **arquivo** é geralmente associado a entidades de vida mais longa, armazenadas em **memória externa**.

Entretanto, esta distinção não é precisa: um arquivo de índices pode ser tratado como uma tabela, enquanto uma tabela de valores de funções pode ser tratada mais como um arquivo

Pesquisa (ou Busca) de Dados

Existe uma variedade enorme de **métodos de pesquisa**.

A escolha do método de pesquisa mais adequado a uma determinada aplicação depende principalmente:

- da **quantidade** dos dados envolvidos,;

- do arquivo estar sujeito a **inserções** e **retiradas** frequentes;

- do **conteúdo** do arquivo ser praticamente estável (neste caso é importante minimizar o tempo de pesquisa, sem preocupação com o tempo necessário para estruturar o arquivo).

Algoritmos de Pesquisa => Tipos Abstratos de Dados

É importante considerar os algoritmos de pesquisa como **tipos abstratos de dados**, com um conjunto de operações associado a uma estrutura de dados, de tal forma que haja uma independência de implementação para as operações.

Operações mais comuns:

1. Inicializar a estrutura de dados.
2. Pesquisar um ou mais registros com determinada chave.
3. Inserir um novo registro.
4. Retirar um registro específico.
5. Ordenar um arquivo para obter todos os registros em ordem de acordo com uma chave.
6. Juntar dois arquivos para formar um arquivo maior.

Pesquisa Sequêncial

É o método de pesquisa mais simples que existe funciona da seguinte forma:

- A partir do **primeiro registro**, pesquise sequencialmente **até encontrar a chave procurada**;

- **se encontrou**, então pare e **retorne o registro**(pesquisa com **sucesso**);

- se chegou no **fim do arquivo** sem encontrar, então retorne erro (pesquisa **sem sucesso**).

Apesar de sua simplicidade, a pesquisa sequencial envolve ideias interessantes, servindo para ilustrar vários aspectos e convenções a serem utilizadas nos outros métodos de pesquisa a serem apresentados.

Pesquisa Sequêncial : Custo Computacional

Anhanguera
Aqui o seu esforço ganha força.

Pesquisa **com sucesso**:

melhor caso : C (n) = 1. (o primeiro do arquivo)

pior caso : C (n) = n. (o último do arquivo)

caso médio : C (n) = (n + 1) / 2. (no meio do arquivo)

Pesquisa **sem sucesso**:

C (n) = n + 1. (não se encontra no arquivo)

O algoritmo de pesquisa sequencial é a **melhor escolha** para o problema de pesquisa em tabelas com **poucos registros**.

Pesquisa Sequêncial : Lista Estática

Como exercício, vamos trabalhar com o conceito de **Pesquisa Sequencial** numa **Lista Estática**, preenchida aleatoriamente.

Operações a serem realizadas:

1 - **Criar** a estrutura do tipo **Lista**;

2 - **Inicializar** a estrutura de dados do tipo Lista ;

3 - Realizar **testes de limites**;

4 - **Preencher** a Lista ;

5 - Função auxiliar para a **impressão** da Lista.

6 - **Pesquisar** um ou mais registros com determinada chave;

Pesquisa Sequêncial : Exemplo no DevC++

Observações:

- Cria um novo projeto:

    - Console Application;

    - C++ Project;

    - Salvar numa pasta utilizável (que você possua permissão de escrita).

Bibliotecas e Estrutura de Trabalho

Anhanguera
Aqui o seu esforço ganha força.

```
#include <cstdlib>
#include <iostream>
#include <stdbool.h>
#include <stdlib.h>

using namespace std;

FILE *arquivo;              // ponteiro para acessar o arquivo

#define MaxLista 100        // define o tamanho maximo da lista

struct  Estrutura_Lista{
    int Dados[MaxLista];
    int ultimo; };
```

Função que Inicializa a Lista de Trabalho

```
void Inicializa_Lista(Estrutura_Lista &x)
{
  x.ultimo = -1;
}
```

Função que testa se a Lista está vazia

```
bool Lista_Vazia(Estrutura_Lista &x)
{
 if (x.ultimo == - 1)
  {
   return true;
  }
 else
  {
    return false;
  }
}
```

Função que testa se a Lista está cheia

```
bool Lista_Cheia(Estrutura_Lista &x)
{
 if (x.ultimo ==   (MaxLista - 1))
   {
    return true;
   }
 else
   {
     return false;
   }
}
```

Função que preenche a Lista com valores aleatórios


```
bool Preencher(Estrutura_Lista &x)
{

 while(Lista_Cheia(x)== false)
  {
     x.ultimo = x.ultimo + 1;
     x.Dados[x.ultimo] = rand() % 1000;
  }
}
```

Função que imprime a Lista preenchida

```
void Imprimir_Lista(Estrutura_Lista &x)
{
     int i;
     if (Lista_Vazia(x)==false)
      {
        i=0;
        for(i=0;i<= x.ultimo; i++)
         {
           printf("\n  Valor %i na posicao %i   ",x.Dados[i],i);
         }
      }
      else
      {
          printf("\n Erro!  Lista Vazia ");
      }
}
```

Função que realiza a pesquisa sequencial

```
bool Pesquisa_Sequencial(Estrutura_Lista &x, int &y)
{
  int i =0;

 while ((i<= MaxLista -1))
  {
     if ( x.Dados[i] == y)
     {
      return true;
     }
    else
     {
      i++;
     }
  }
   return false;
}
```

Teste da Lista criada
Realização da Pesquisa Sequencial

Anhanguera
Aqui o seu esforço ganha força.

```
int main(int argc, char *argv[])        //  função principal
{

     Estrutura_Lista   Lista1;          // criação da Lista

     Inicializa_Lista(Lista1);          // inicialização da Lista

     Preencher(Lista1);                 // preenchimento da Lista

     Imprimir_Lista(Lista1);            // impressão do conteúdo da Lista
```

... Continuação...

```
int chave;                  // variável auxiliar para a chave

printf("\n  Digite a chave a ser pesquisada : ");

scanf("%d",&chave);         // entrada via teclado da chave a ser pesquisada
```

... Continuação...

```
/* chama a função de pesquisa com a chave lida do teclado

e retorna verdadeiro ou falso,  se a chave existir na Lista ou não */


if(Pesquisa_Sequencial(Listal, chave)==true)
{
  printf("\n  Sucesso - Chave Encontrada ");
}
else
{
  printf("\n Sem sucesso - Chave não Encontrada ");
}
```

... Continuação

```
    printf("\n");              // salta uma linha

    system("PAUSE");           // pausa no programa

    return EXIT_SUCCESS;       // retorna o controle ao Sistema Operacional


} // fim da função principal ( main )
```

Exercícios:

1- Modificar a função Pesquisa_Sequencial( ) para que a mesma retorne:

- a **posição** na Lista, do elemento que corresponde a chave procurada, se encontrado;

- 1 se a chave não foi encontrada;

2 – Criar uma função para retirar um elemento da Lista. A função deve retornar:

- verdadeiro (**true**), se elemento encontrado e excluído;

- falso (**false**), se elemento não se encontra na Lista.

Enviar até 01/03/2016 via email cp_1sem_2016@pedraorc.com.br

Bibliografia da apresentação:

ZIVIANI, N. ***Projeto de Algoritmos com implementações em Pascal e C***. 2ª edição. São Paulo: Pioneira Thomson Learning, 2005. 552 p. Disponível em http://www2.dcc.ufmg.br/livros/algoritmos/

http://codificandooo.blogspot.com.br/2013/03/java-tipos-primitivos.html

Mizrahi, Victorine Viviane – **Programação Estruturada** – PLT 193

Tanenbaum, Aarom M. – **Estrutura de dados** – PLT 194

Plano de Ensino e Aprendizagem (PEA)

Anhanguera
Aqui o seu esforço ganha força.

Bibliografia Básica Padrão
1) ZIVIANI, Nivio. **Projeto de algoritmos : com implementações em Pascal e C**. 2ª ed. São Paulo: Pioneira - Thomson Learning, 2007.

Bibliografia Básica Unidade: Faculdade Anhanguera de Rio Claro (FRC)
1) SANTOS, Clésio Saraiva dos; AZEREDO, Paulo Alberto de. **Tabelas: organização e pesquisa.**. 1ª ed. Porto Alegre: Sagra Luzzatto, 2001.
2) CORMEN, Thomas H.. **Algoritmos** : Teoria e Prática. 2ª ed. Rio de Janeiro: Campus - Elsevier, 2002.

Bibliografia Complementar: Faculdade Anhanguera de Rio Claro (FRC)
1) TENENBAUM, Aaron M. **Estrutura de Dados Usando C**. 1ª ed. São Paulo: Pearson, 2009.
2) CELES, Waldemar. **INTRODUÇÃO A ESTRUTURAS DE DADOS COM TÉCNICAS DE PROGRAMAÇÃO EM C**. 4ª ed. Rio de Janeiro: Campus - Elsevier, 2004.
3) SZWARCFITER, Jaime Luis; MARKENZON, Lilian. **Estruturas de Dados e Seus Algoritmos**. 2ª ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 1994.
4) SCHILDT, Herbert. **C Completo e Total**. 3ª ed. São Paulo: Pearson, 2005.
5) PEREIRA, Caio Mário da Silva. **Estrutura de Dados Fundamentais** : Conceirtos e Aplicações. 12ª ed. São Paulo: Érica, 2008.