Estrutura de Dados – 1º semestre de 2020

Professor Mestre Fabio Pereira da Silva

Introdução

- Estruturas de dados são objetos que armazenam dados de forma eficiente, oferecendo certos “serviços” para o usuário (ordenação eficiente dos dados, busca por meio de palavras chave, etc).
- Técnicas de programação orientada a objetos são úteis quando temos que codificar estruturas de dados.
- As estruturas básicas principais são:
  - Listas, pilhas e filas estáticas e dinâmicas
  - Árvores binária
  - Grafos.
  - Hash tables

Introdução

Problema
Manipular um conjunto de fichas de um fichário.

Solução
Organizar as fichas em ordem alfabética

Operações possíveis
Inserir ou retirar um ficha, procurar uma ficha, etc.

Estrutura de Dados Correspondente
LISTA – seqüência de elementos dispostos em ordem.

Introdução

Problema

Organizar as pessoas que querem ser atendidas num guichê.

Solução

Colocar as pessoas em fila.

Operações possíveis

À medida que uma pessoa é atendida no guichê, outra entra no final da fila... Não é permitido "furar" a fila, ou seja, entrar uma pessoa entre outras que já estão presentes.

Estrutura de Dados Correspondente

FILA – seqüência de elementos dispostos em ordem com uma regra para a entrada e saída dos elementos (o primeiro que chega também é o primeiro que sai da estrutura).

Introdução

Problema

Organizar um conjunto de pratos que estão sendo lavados, um a um, em um restaurante.

Solução

Colocar os pratos empilhados.

Operações possíveis

Colocar um prato limpo no alto da pilha, retirar um prato do alto da pilha, etc...

Estrutura de Dados Correspondente

PILHA – seqüência de elementos dispostos em ordem, mas com uma regra para entrada e saída dos elementos (o último que chega é o primeiro que sai da estrutura).

Introdução

Problema

Conseguir um modo de visualizar o conjunto de pessoas que trabalham em uma empresa, tendo em conta sua função.

Solução

Construir um organograma da empresa.

Operações possíveis

Inserir ou retirar certas funções, localizar uma pessoa, etc...

Estrutura de Dados Correspondente

ÁRVORE – estrutura de dados que caracteriza uma relação de hierarquia entre os elementos (uma pessoa não pode pertencer a dois departamentos diferentes, cada diretoria tem os seus próprios departamentos, etc.).

Introdução

Problema

Estabelecer um trajeto para percorrer todas as capitais de um país.

Solução

Utilizar um mapa que indique as rodovias existentes e estabelecer uma ordem possível para percorrer todas as cidades.

Operações possíveis

Encontrar um modo de percorrer todas as cidades, determinar o caminho mais curto para ir de uma cidade para outra, etc.

Estrutura de Dados Correspondente

GRAFO – estrutura bastante genérica que organiza vários elementos, estabelecendo relações entre eles, dois a dois.

Introdução

- Quais informações manipulamos diariamente como uma sequência?
- Alunos
- Cinemas
- Contatos pessoais
- Contatos profissionais
- Escolas
- Livros
- Presentes

Listas

- Listas são conjuntos de elementos, objetos, variáveis, tarefas, ou qualquer coisa que se possa enumerar e formar um conjunto;
- As listas estão presentes em nossa vida, desde o nosso nascimento, por exemplo, com a lista de compras que nossos pais tiveram que fazer para nós.

Listas

- Exemplo de Lista de Compras:
  - 5Kg de farinha;
  - 2Kg de açucar;
  - 500g de carne moída;
  - 2Kg de arroz;
  - 4L de leite;
  - 1Kg de feijão;

Listas

- Exemplo de Lista Telefônica:
  - Asdf de Zxcv: (44) 4444-4444
  - Beutrano Cruz: (33) 3333-3333
  - Ciclano da Silva: (22) 2222-2222
  - Fulano de Tal: (11) 1111-111

Listas

- Uma estrutura que armazena elementos de forma alinhada, ou seja, com elementos dispostos um após o outro.
- São estruturas lineares que armazenam vários elementos de um mesmo tipo.
- Podem ser adequadas quando não é possível prever a demanda por memória, permitindo a manipulação de quantidades imprevisíveis de dados, de formato também imprevisível.

Listas

- Definição dada por Knuth para uma lista linear : “Uma lista linear X é um conjunto de nodos X(1), X(2), ….,X(n), tais que:
  - a) X(1) é o primeiro nodo da lista;
  - b) X(n) é o último nodo da lista; c)
  - c) Para 1<k
- Ou, mais simplesmente, “Uma lista linear é a estrutura de dados que permite representar um conjunto de dados de forma a preservar a relação de ordem linear entre eles.”

Listas

- Sequência de zero ou mais itens x 1; x 2; ... ; x n, na qual xi é de um determinado tipo e n representa o tamanho da lista linear.
- Sua principal propriedade estrutural envolve as posições relativas dos itens em uma dimensão.
  - Assumindo n>= 1, x 1 é o primeiro item da lista e x n é o último item da lista.
  - xi precede xi+1 para i = 1; 2; ... ; n - 1
  - xi sucede xi-1 para i = 2; 3; ... ; n – o elemento xi é dito estar na i-ésima posição da lista.

Listas

- Uma lista é uma sequência ordenada de elementos do mesmo tipo. Por exemplo, um conjunto de fichas de clientes de uma loja, organizadas pela ordem alfabética dos nomes dos clientes.
- Neste fichário é possível introduzir uma nova ficha ou retirar uma velha, alterar os dados de um cliente etc.
- Do ponto de vista matemático, uma lista é uma sequência de zero ou mais elementos de um determinado tipo.
- Geralmente se representa uma lista de elementos, separando-os por vírgulas. a1 , a2 , ... na onde n >= 0 e ai é um elemento da lista.
- O número n é dito comprimento da lista. Se n=0 temos uma lista vazia.

Operações

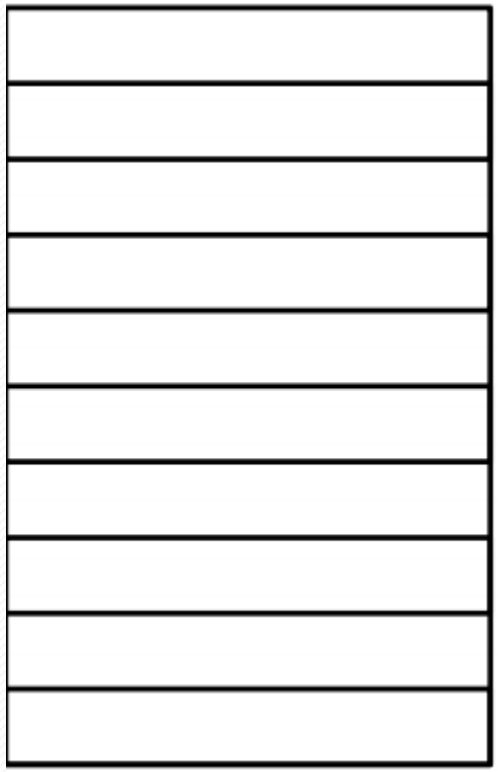
- Criação de uma lista
- Inserção
- Ordenação
- Remoção
- Busca
- Concatenar duas listas
- Determinar o número de nós de uma lista
- Cópia de uma lista
- Intercalação
- Destruição de uma lista

Implementação de listas lineares

- Há varias maneiras de implementar listas lineares.
- Cada implementação apresenta vantagens e desvantagens particulares.
- Vamos estudar duas maneiras distintas
  - Usando alocação sequencial e estática (com vetores).
  - Usando alocação não sequencial e dinâmica (com ponteiros): Estruturas Encadeadas.

Listas em Arrays

- Imagine que a lista anterior tinha posições fixas e prédeterminadas:
- Um array é uma estrutura com posições fixas, cada elemento da lista deve ser colocado em uma posição no array;
- Ao inserir ou excluir um elemento, talvez seja necessário realocar todos os demais elementos.



Lista Estática

- Prós:
  - Criar um array de qualquer tamanho é muito simples;
  - Não há necessidade de compreender ponteiros ou referências;
- Contras:
  - Limitações quanto ao tamanho de memória;
  - Custo computacional maior;
  - Alocação de memória exagerada.

Lista Encadeada

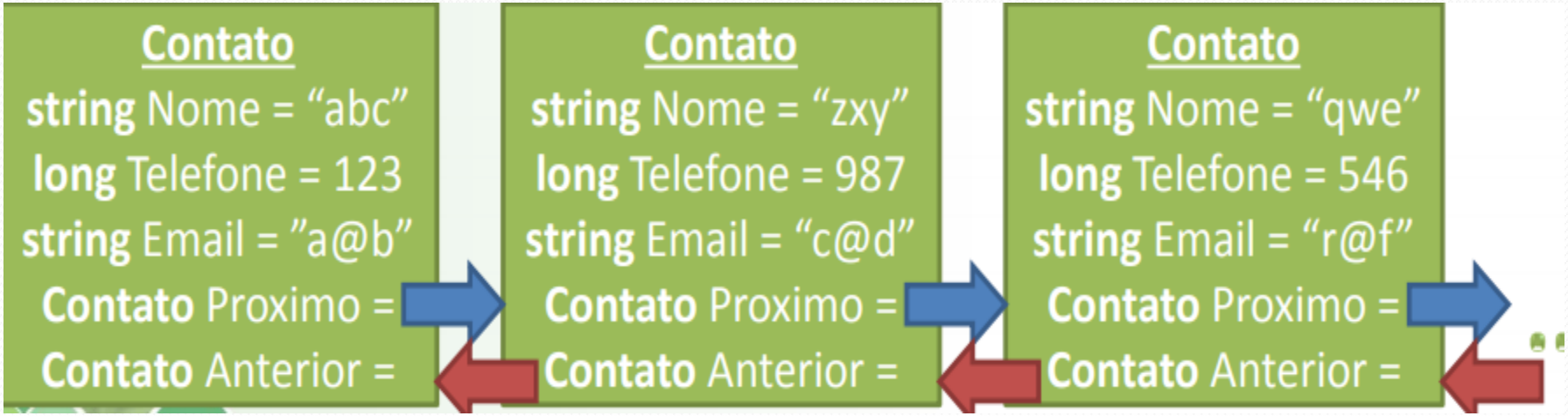
- Encadeado, Dicionário Houaiss:
- – adjetivo
- – 1. disposto ou ligado por ou como por cadeias; ordenado, junto;
- – 2. preso, submetido;

Lista Encadeada

- Prós:
  - Extremamente eficiente no custo de memória e de processamento;
  - Evita a movimentação de todos os elementos;
- Contras:
  - Envolve conceitos mais avançados de programação como ponteiros ou referências;
  - Dificuldade de implementação.

Lista Encadeada

Lista Encadeada



Alocação de Memória

- Alocação estática → Variável alocada ocupa espaço fixo e contíguo na memória;

- Alocação dinâmica → Variável alocada ocupa espaço variável e é criada segundo a necessidade do programa.

Vantagens e Desvantagens da Alocação Dinâmica

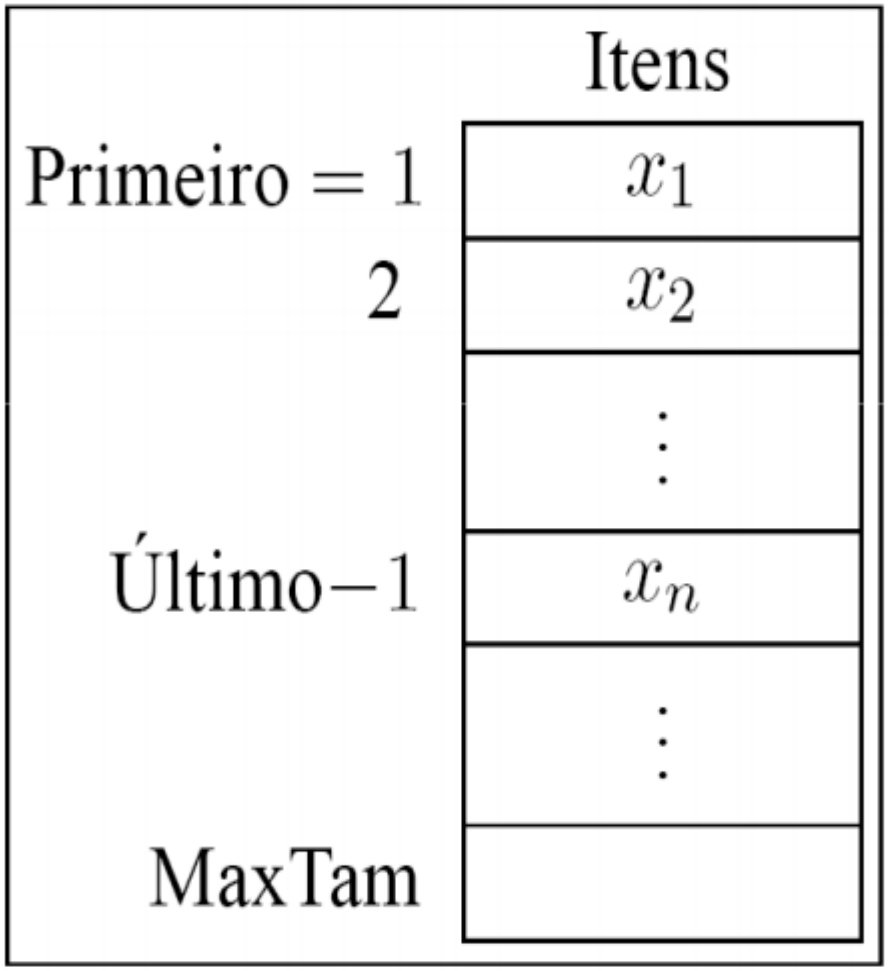
- Se alocamos dinamicamente, podemos aumentar e diminuir o tamanho de nossa estrutura quando quisermos!

- Entretanto, necessitaremos de mais algumas operações para buscar, inserir e/ou remover informações;

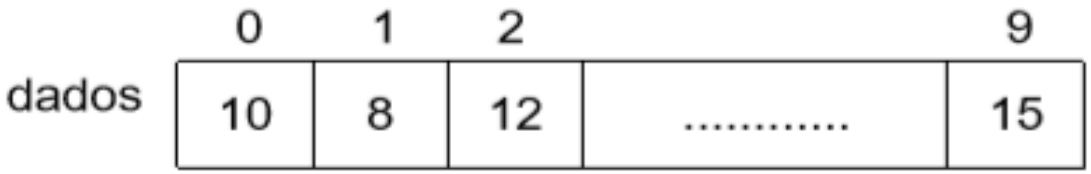
- Além disso, um array (estático) de vinte posições geralmente ocupa menos espaço que uma lista cujos elementos foram criados um a um dinamicamente.

Lista Estática x Lista Dinâmica

Alocação Estática	Alocação Dinâmica
Quantidade constante de elementos	Não há quantidade máxima de elementos (o limite é a memória do computador)
Aloca espaço de acordo com a quantidade de elementos	Utiliza somente o espaço de memória suficiente
Usa arrays	Utiliza ponteiros para indicar a posição de memória que o endereço inserido na lista será armazenado


Itens
Primeiro = 1
x_1
2
x_2
Último−1
x_n
MaxTam

Dada a Lista



Lista
- dados: int[] - tamanho: int
+ Lista() + vazia(): boolean + cheia(): boolean + adicionaInicio(e: int): void + adicionaFinal(e: int): void + adiciona(int e, int posicao): void + removeInicio(): int + removeFinal(): int + remove(int posicao): int + obtemPrimeiro(): int + obtemUltimo(): int

ALOCAÇÃO ESTÁTICA

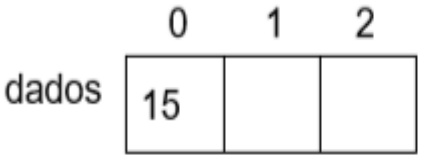
Dada a Lista Vazia

Adicione o elemento 15 no início da lista, processo:

- Lista está cheia?
- Não:
  - guarde 15 no vetor denominado dados, índice 0
  - some 1 em tamanho

ALOCAÇÃO ESTÁTICA

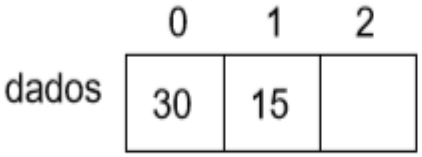
Dada a Lista



tamanho: 1

Adicione o elemento 30 no início da lista, processo:

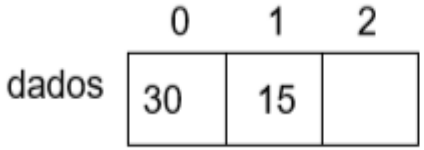
- Lista está cheia?
- Não:
  - passe, a partir do último elemento, todos os elementos uma posição à frente
  - guarde 30 no vetor denominado dados, índice 0
  - some 1 em tamanho



tamanho:2

ALOCAÇÃO ESTÁTICA

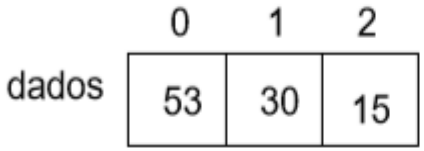
Dada a Lista



tamanho: 2

Adicione o elemento 53 no início da lista, processo:

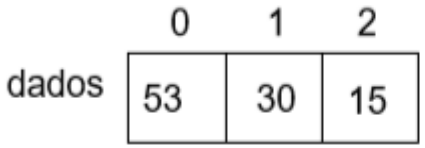
- Lista está cheia?
- Não:
  - passe, a partir do último elemento, todos os elementos uma posição à frente
  - guarde 53 no vetor denominado dados, índice 0
  - some 1 em tamanho



tamanho:3

ALOCAÇÃO ESTÁTICA

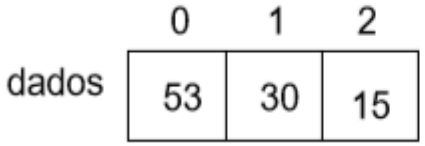
Dada a Lista



tamanho: 3

Adicione o elemento 47 no início da lista, processo:

- Lista está cheia?
  - Sim:
    - Mostre a mensagem “Lista cheia”



tamanho:3

Exemplo de Lista com números inteiros positivos

```
/**
Classe para organização de números inteiros em lista
*/
public class ListaDeInteiros {
        /** array de inteiros */
       private int[] dados;
        /** quantidade de elementos guardados no
            array*/
       private int tamanho;

        /**
            Método construtor
        */
        public ListaDeInteiros(){
            dados = new int[10];
            tamanho = 0;
        }
```

ALOCAÇÃO ESTÁTICA

Exemplo de Lista com números inteiros positivos

```
/**
      Método que  verifica se a lista está vazia
      @return  true se está vazia e false se há pelo menos
      um elemento
 */
public boolean vazia(){
    if (tamanho == 0){
      return true;
      }
    return false;
}
```

ALOCAÇÃO ESTÁTICA

Exemplo de Lista com números inteiros positivos

```
/**
      Método que  verifica se a lista está cheia
      @return  true se está cheia e false se há pelo menos
      uma posição vazia
*/
public boolean cheia(){
    if (tamanho == elemento.length){
        return true;
    }
    return false;
}
```

ALOCAÇÃO ESTÁTICA

```
/**
Método que adiciona o elemento no final da lista
@param n novo elemento
*/
public void adicionaFinal(int n) {
          if (!cheia()){
               dados[tamanho] = n;
               tamanho++;
           }
           else
           {
               System.out.println("Lista Cheia!");
           }
    }
```

ALOCAÇÃO ESTÁTICA

```
/**
Método que remove o elemento do final da lista e
o retorna
@return elemento removido
*/
    public int removeFinal ( ){
        int n=-1;
        if (!vazia()){
            tamanho--;
            n = dados[tamanho];
        }   else
      {
          System.out.println("Lista Vazia!");
      }
        return n;
    }
}
```

ALOCAÇÃO ESTÁTICA

Exemplo de Lista com RA e Nome de Alunos:

```
/**
Lista de alunos
*/
public class ListaDeAlunos {
        /** objeto da classe Aluno*/
        private Aluno[] dados;
        /** total de alunos adicionados à
            lista*/
        private int totalDeAlunos;

        /**
        Método construtor
        */
        public ListaDeAlunos(){
            dados = new Aluno[3];
            totalDeAlunos = 0;
        }
```

ALOCAÇÃO ESTÁTICA

```
/**
     Método que verifica se a lista está
     vazia
     @return true se a lista está vazia e
     false se possui pelo menos um objeto
*/
public boolean vazia(){
     if (totalDeAlunos == 0){
        return true;
   }
     return false;
}
```

ALOCAÇÃO ESTÁTICA

```
/**
     Método que verifica se a lista está
     cheia
     @return true se a lista está cheia e
     false se possui pelo menos uma
     posição livre
*/
public boolean cheia(){
     if (totalDeAlunos == dados.length){
          return true;
     }
     return false;
}
```

ALOCAÇÃO ESTÁTICA

```
/**
Método que adiciona elemento no final da lista
@param aluno instância da classe Aluno que
será adicionado à lista
*/
public void adicionaFinal(Aluno aluno) {
     if (!cheia()){
       dados[totalDeAlunos] = aluno;
       totalDeAlunos++;
     } else
     {
       System.out.println("Lista Cheia!");
     }
}
```

ALOCAÇÃO ESTÁTICA

```
    /**
    Método que remove  e retorna  o elemento no
final da lista
    @return aluno instância da classe Aluno que
foi removido da lista
    */

    public Aluno removeFinal ( ){
        Aluno aluno;
        if (!vazia()){
            totalDeAlunos--;
            aluno = dados[totalDeAlunos];
        }
        return aluno;
    }

}
```

Contatos

- Email: fabio.silva321@fatec.sp.gov.br
- Linkedin: https://br.linkedin.com/in/b41a5269
- Facebook: https://www.facebook.com/fabio.silva.56211