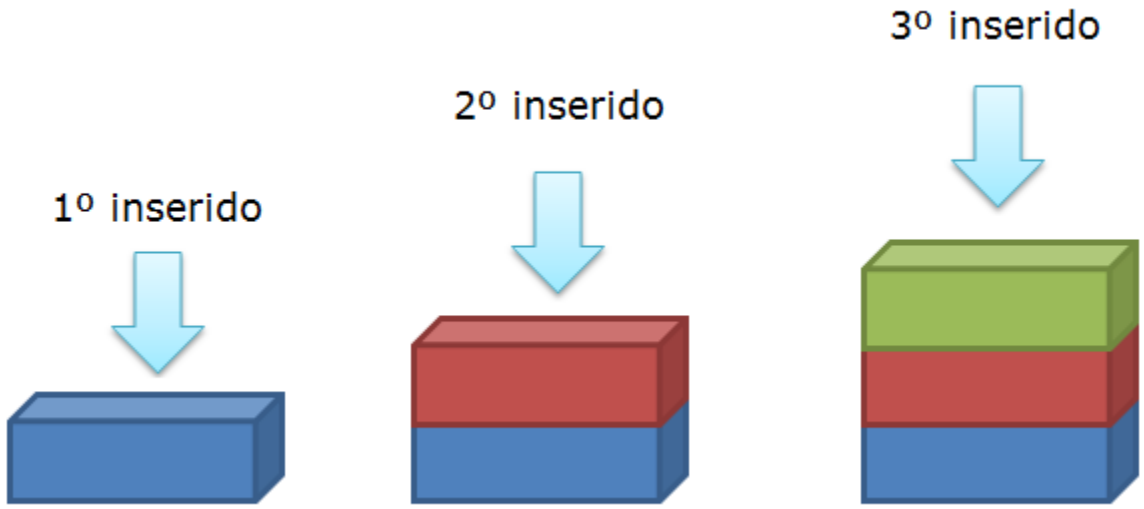
Estrutura e Banco de Dados

Claudiney Sanches Júnior

FILO (First-In, Last-Out)

- A pilha implementa o conceito de FILO (First-In, Last-Out) ou “Primeiro a Entrar, Último a Sair”.
  - o último elemento a ser inserido na pilha é o primeiro a ser removido
  - enquanto o primeiro a ser inserido é o último que sai.
- Imagine que você deve organizar vários livros que encontram-se espalhados para depois distribuí-los ordenadamente. Primeiramente você juntará todos os livros em uma pilha, um sobre o outro.

FILO (First-In, Last-Out)

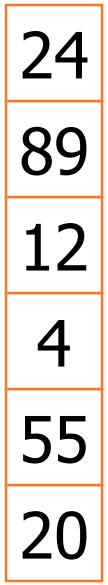


Remover

- Feito isso, você passará a distribuí-los um a um.
- O primeiro livro a ser removido da pilha para ser entregue é aquele que se encontra no topo, ou seja, o último que foi inserido.

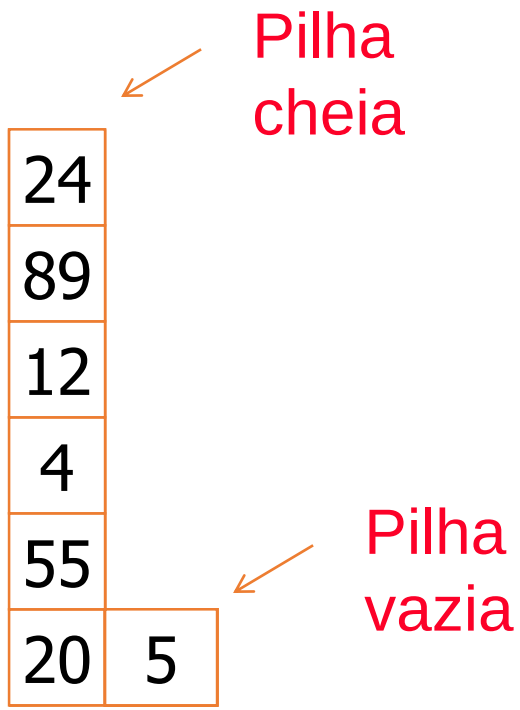
Remover

Pilhas



A Pilha é uma estrutura de dados cujo funcionamento é inspirado no de uma pilha “natural”.

Pilhas usando Vetores



Vetores possuem um espaço limitado para armazenar dados;

necessitamos definir um espaço grande o suficiente para a nossa pilha;

necessitamos de um indicador de qual elemento do vetor é o atual topo da pilha.

Modelagem da Pilha

Aspecto Funcional:

colocar e retirar dados da pilha;

testar se a pilha está vazia ou cheia;

Colocar e retirar dados da pilha:

- Empilha(dado) / Push(dado)
- Desempilha(dado) / Pop(dado)
- Topo

Testar se a pilha está vazia ou cheia:

- PilhaCheia
- PilhaVazia

Inicializar ou limpar:

- InicializaPilha

FILO

- A nossa classe FILO possuirá apenas três métodos:
- **Push(obj)**:
  - insere o elemento recebido como parâmetro no final da lista
- **Pop**():
  - retorna o valor do último elemento inserido e o remove da lista
  - este método é o mais utilizado nesse contexto
- **Ler** ():
  - retorna o valor do último elemento inserido, o do topo da pilha.

```
function FILO(){
    this.pilha = new Array();
    this.Push = function(obj){
        this.pilha[this.pilha.length] = obj;
    }
    this.Pop = function(){
        if(this.pilha.length > 0){
            var obj = this.pilha[this.pilha.length - 1];
            this.pilha.splice(this.pilha.length - 1,1);
            return obj;
        }else{
            alert("Não há objetos na pilha.")
        }
    }
```

```
    this.Ler = function(){
        if(this.pilha.length > 0){
            return this.pilha[this.pilha.length - 1];
        }else{
            alert("Não há objetos na pilha.")
        }
    }
}
```

Testando as classes

- Vamos então realizar alguns testes com as classes criadas, para demonstrar o funcionamento delas na prática.
- Nos exemplos, as classes foram salvas nos arquivos FILO.js mantido no mesmo diretório do arquivo HTML

```
<html>
<head>
    <script type="text/javascript" src=“FILO.js"></script>
    <script type="text/javascript"/>
        var pilhaCont = new FILO();
        pilhaCont.Push(1);
        pilhaCont.Push(2);
        pilhaCont.Push(3);
        var item = pilhaCont.Pop();
        alert(item);
    </script>
</head>
<body>
    <h1 id="txt"></h1>
</body>
</html>
```

FIFO

- Fila ou Queue como também é conhecida
- é uma estrutura de dados que implementa o conceito de
  - FIFO (First-In, First-Out) ou “Primeiro a Entrar, Primeiro a Sair”.

A Fila é uma estrutura de dados que simula uma fila da vida real.

Possui duas operações básicas:

- incluir no fim da fila;
- retirar do começo da fila;

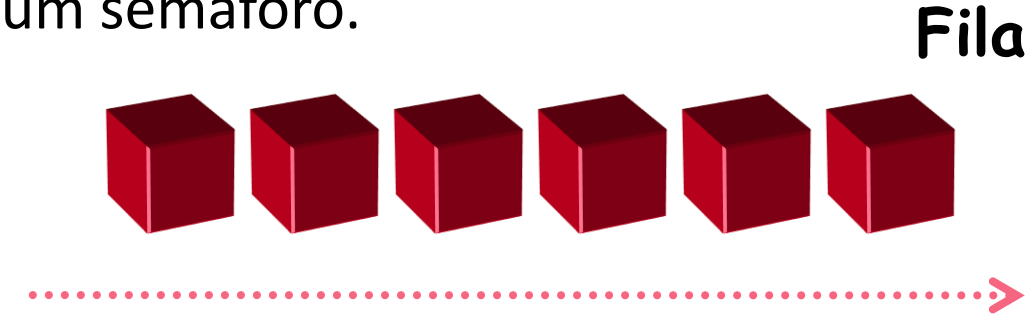
Fila

- Essa situação é semelhante, por exemplo a uma fila de supermercado ou banco, onde o primeiro usuário a entrar na fila é também o primeiro a ser atendido e sair dela. Nesse caso, ilustrações são dispensadas pois trata-se de um conceito ainda mais simples que o anterior.

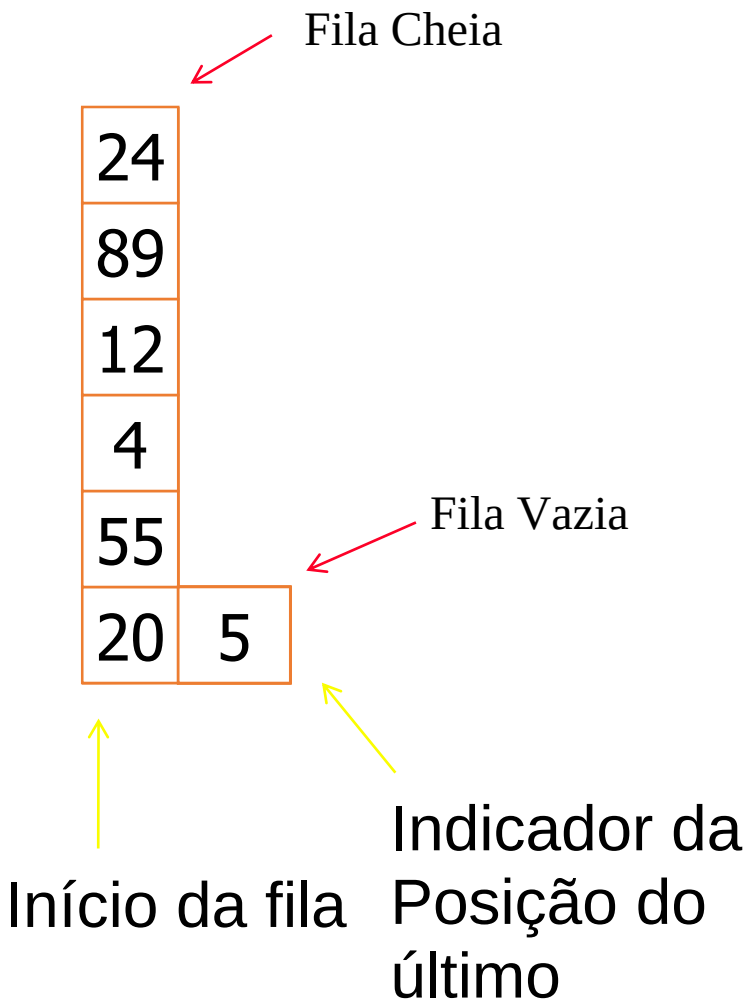
Filas

É uma estrutura de dados importantíssima para gerência de dados/processos por ordem cronológica.

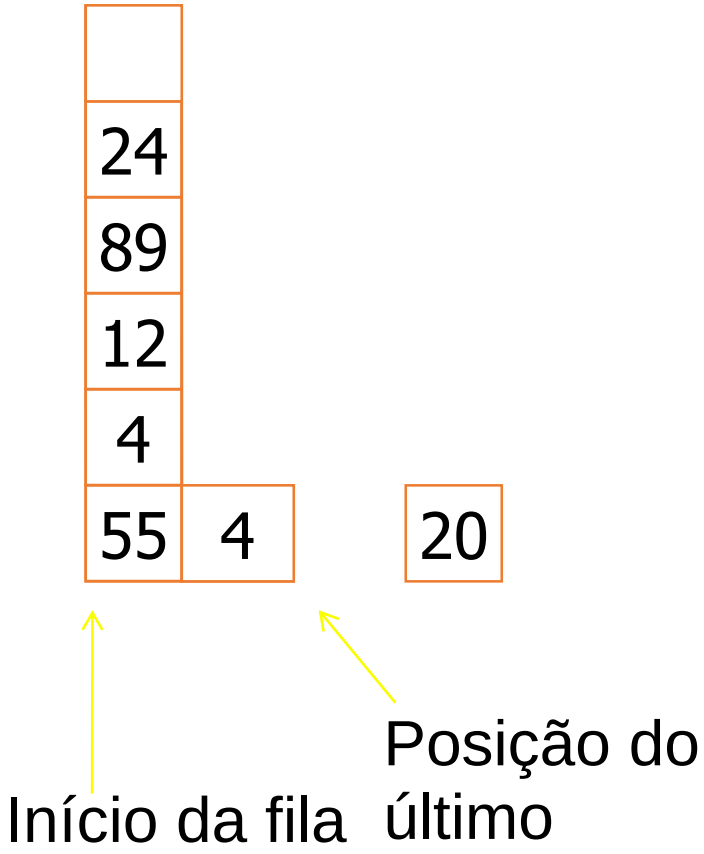
São exemplos de implantação: Fila de impressão; Fila de pedidos de uma expedição, simulação de processos seqüenciais, fila de camisetas a serem estampadas, simulação de fluxo de um caixa de supermercado, simulação de um cruzamento com um semáforo.



Filas usando Vetores



Algoritmo Retira



Procedimento:

testamos se há elementos;

decrementamos o fim da fila (último);

salvamos o primeiro elemento em variável auxiliar;

empurramos tudo para a frente.

Parâmetros:

fila (global).

Modelagem da Fila com Vetor

Aspecto Funcional:

colocar e retirar dados da fila;

testar se a fila está vazia ou cheia;

Colocar e retirar dados da fila:

- Inclui(dado)
- Retira

Testar se a fila está vazia ou cheia:

- FilaCheia
- FilaVazia

Inicializar ou limpar:

- InicializaFila

Fila

- Nesta estrutura temos dois métodos principais
  - um para inserir um item na fila
  - outro para ler e remover o primeiro elemento
- Vale salientar que este segundo método sempre retornará o elemento de índice zero, ou seja, o primeiro.

Fila

- De forma complementar, a classe que será apresentada aqui terá um método a mais, que permitirá a leitura do primeiro elemento sem que este seja removido da coleção. Em suma, os métodos são os seguintes:
  - **Inserir(obj):** insere um item no final da fila.
  - **RemoverPrimeiro**: retorna o valor do primeiro elemento da fila, o de índice zero, e o remove.
  - **LerPrimeiro**: retorna o valor do primeiro elemento da lista, mas sem que este seja removido.

Classe FIFO

```
function FIFO(){
    this.fila = new Array();
    this.Inserir = function(obj){
        this.fila[this.fila.length] = obj;
    }
    this.RemoverPrimeiro = function(){
        if(this.fila.length > 0){
            var obj = this.fila[0];
            this.fila.splice(0,1);
            return obj;
        }else{
            alert("Não há objetos na fila.")
        }
    }
```

Classe FIFO

```
    this.LerPrimeiro = function(){
        if(this.fila.length > 0){
            return this.fila[0];
        }else{
            alert("Não há objetos na fila.")
        }
    }
}
```

Lista - Definição

- Uma Lista é um conjunto de dados dispostos e/ou acessáveis em uma sequência determinada.
  - Este conjunto de dados pode possuir uma ordem intrínseca (Lista Ordenada) ou não
  - Este conjunto de dados pode ocupar espaços de memória fisicamente consecutivos, espelhando a sua ordem, ou não
  - Se os dados estiverem dispersos fisicamente, para que este conjunto seja uma lista, ele deve possuir operações e informações adicionais que permitam que seja tratado como tal (Lista Encadeada)

2 Aspectos

- Em um projeto de software, 2 aspectos devem ser considerados:
  - De que forma estão organizados os dados, qual a sua estrutura.
  - Quais procedimentos atuam sobre estes dados, que operações podem ser realizadas sobre eles.
  - Vamos ver agora estes aspectos para as **listas**.

Listas

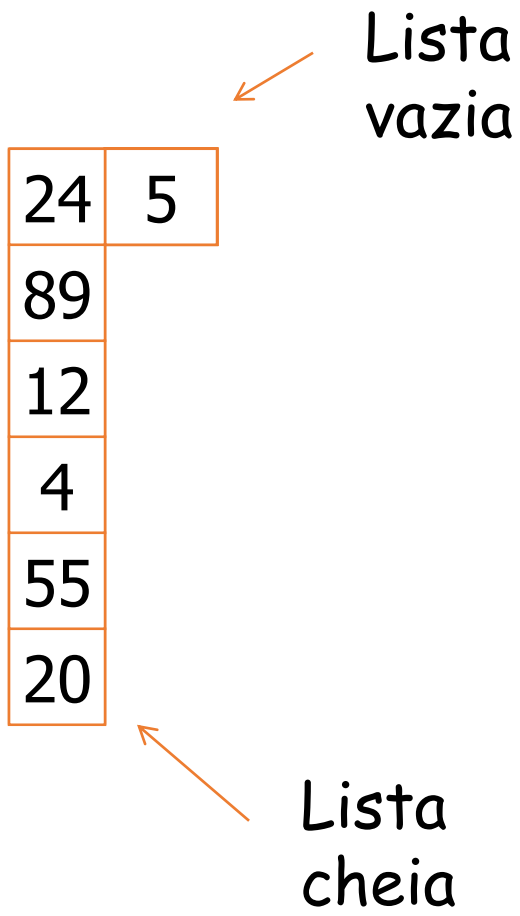
4
12
20
24
55
89

Uma lista pode ser Ordenada ou não.

Quando for ordenada, pode o ser por alguma característica intrínseca dos dados (ex: ordem alfabética).

Pode também refletir a ordem cronológica (ordem de inserção) dos dados).

Listas usando Vetores



Vetores possuem um espaço limitado para armazenamento de dados.

Necessitamos definir um espaço grande o suficiente para a lista.

Necessitamos de um indicador de qual elemento do vetor é o atual ultimo elemento da lista.

Listas

A Lista é uma estrutura de dados cujo funcionamento é inspirado no de uma lista “natural”

Uma lista pode ser Ordenada ou não.
Quando for ordenada, pode o ser por alguma característica intrínseca dos dados (ex: ordem alfabética).
Pode também refletir a ordem cronológica (ordem de inserção) dos dados).

Modelagem de Listas utilizando Programação Estruturada

Modelaremos a Estrutura de Dados Lista utilizando a técnica da Programação Estruturada e como forma de armazenamento um Vetor (Array).

- Veremos somente algoritmos.
- Veremos algoritmos tanto para uma lista cronológica como para uma lista ordenada.

Modelagem da Lista

Aspecto Estrutural:

- Necessitamos de um vetor para armazenar as informações.
- Necessitamos de um indicador da posição atual do ultimo elemento da lista.
- Necessitamos de uma constante que nos diga quando a lista está cheia e duas outras para codificar erros.

Modelagem da Lista

Aspecto Funcional:

- Colocar e retirar dados da lista.
- Testar se a lista está vazia ou cheia e outros testes.
- Inicializa-la e garantir a ordem dos elementos.

Modelagem da Lista

Operações: Colocar e retirar dados da lista:

- Adiciona(dado)
- AdicionaInício(dado)
- AdicionaPosição(dado,posição)
- Retira()
- RetiraDoInício()
- RetiraDaPosição(posição)

```
// JavaScript Document
function Lista(){
    this.mlist = new Array();
    this.Inserir = function(obj){
            this.mlist[this.mlist.length]=obj;
    }
    this.InserirPos = function(obj, pos){
      pos--;
      if(pos<this.mlist.length)
          this.mlist.splice(pos,0,obj);
      else
         alert("Posição invalida");
    }
```

```
this.RemoverPrimeiro = function(){
        if(this.mlist.length>0){
                this.mlist.splice(0,1);
        } else {
                alert("Não há objetos na Lista");
        }
}
this.RemoverPos = function(pos){
        if(pos<(this.mlist.length+1))
                this.mlist.splice((pos-1),1);
}
```

```
        this.Escrever = function(){
                if(this.mlist.length>0){
                        for(co=0;co<this.mlist.length;co++)
                                document.write(this.mlist[co]+"</br>");
                }else{
                        alert("Não há objetos na Lista");
                }
        }
}
```

```
<!doctype html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Convidados Churras</title>
      <script src="Lista.js"></script>
      <script>
       var Convidados = new Lista();
       Convidados.Inserir("Maria");
       Convidados.Inserir("Elisangela");
       Convidados.Inserir("Rebecca");
       Convidados.Inserir("Thais");
       Convidados.Inserir("Claudiney");
       Convidados.RemoverPos(1);
       Convidados.InserirPos("Pedro",5);
       Convidados.Escrever();
      </script>
   </head>
   <body>
   </body>
</html>
```