Estácio

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ESTRUTURA DE DADOS

AULA 10

RICARDO EIJI KONDO, Me.

TÓPICOS

UNIDADE IV - Listas, Pilhas e Filas

4.1 - Fundamentos dos Tipos Abstratos de Dados (TAD)

4.1.1 - TADs Lineares (pilhas, filas, listas, sequencias, dicionários, tabelas de hash, filas de prioridade, etc.)

4.1.2 - TADs Não-Lineares: (árvores e grafos)

4.2 - Lista

4.2.1 - Listas Sequenciais

4.2.2 - Listas Encadeadas

4.2.2.1 - Listas Simplesmente Encadeadas

4.2.2.2 - Listas Duplamente Encadeadas

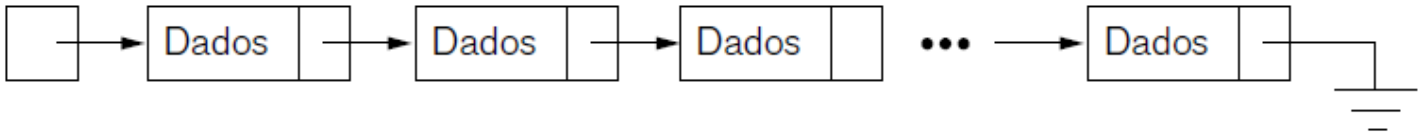
4.2.2.3 - Listas Circulares Simples

4.3 - Pilha

4.4 - Fila

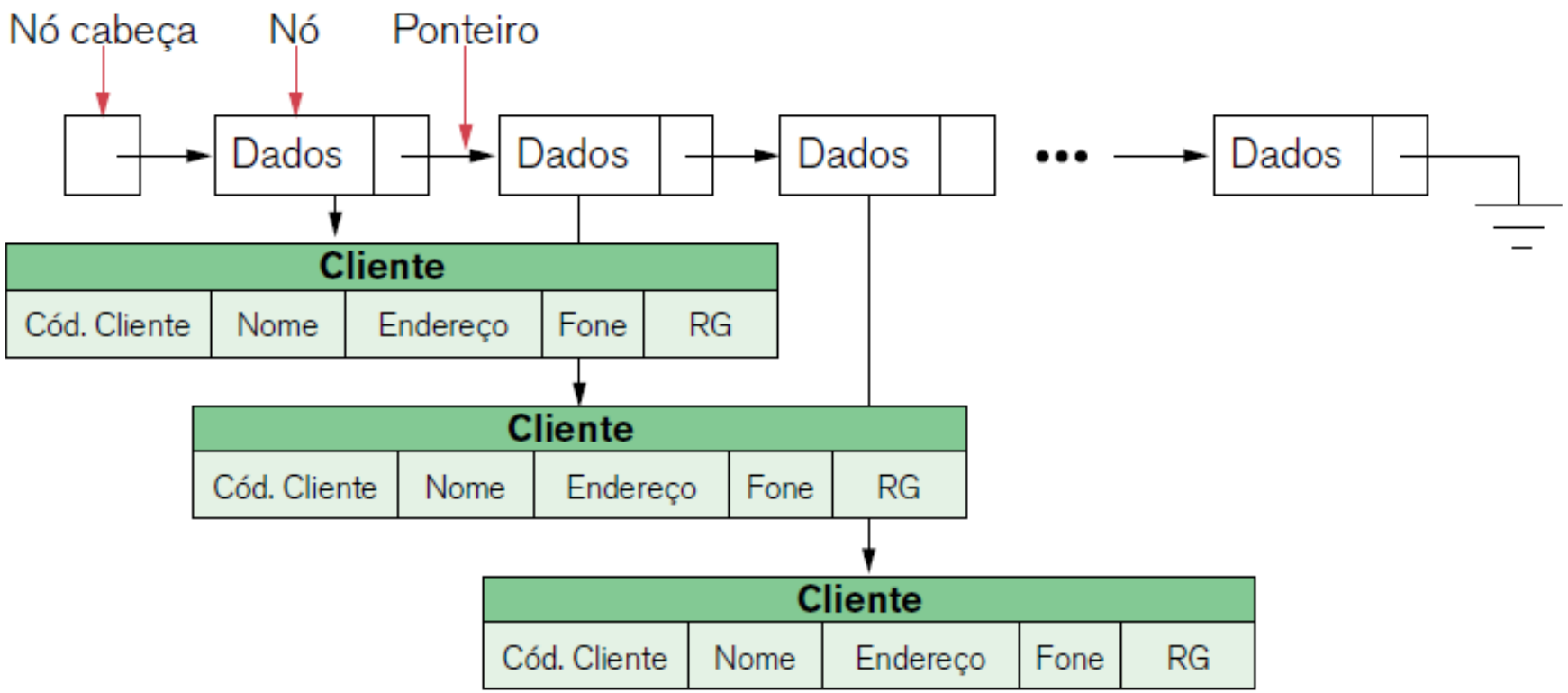
LISTA LINEAR ENCADEADA

- Nas listas lineares encadeadas não há a necessidade de pré-determinar a quantidade máxima de elementos da lista. Os elementos não são armazenados de forma contígua e ocupam qualquer posição de memória disponível.



- A ligação entre os elementos dispersos na memória são feitos através de ponteiros. Ou seja, cada nó, deve conter, além dos registros, uma referência para o próximo nó da lista.
- A quantidade máxima de elementos é determinada pela quantidade de memória disponível que pode ser alocada pelo programa. Desta forma temos uma estrutura dinâmica, ou seja, a alocação de memória é feita durante a execução do programa.

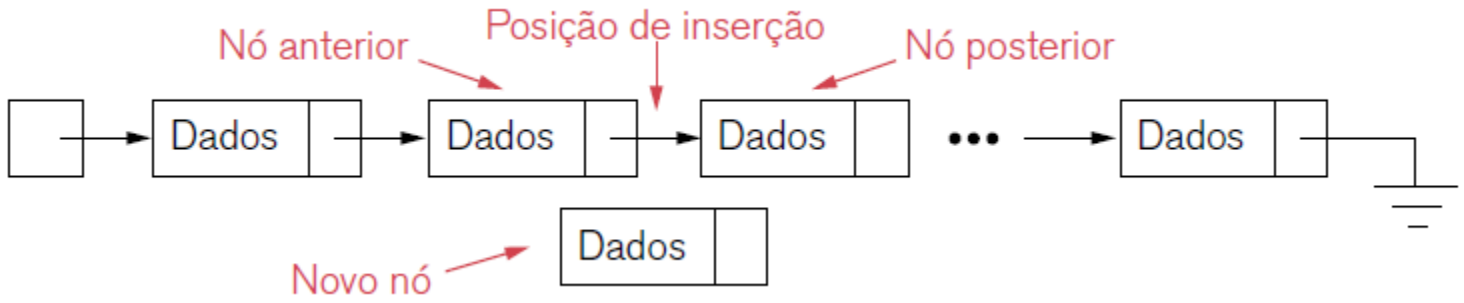
LISTA LINEAR ENCADEADA



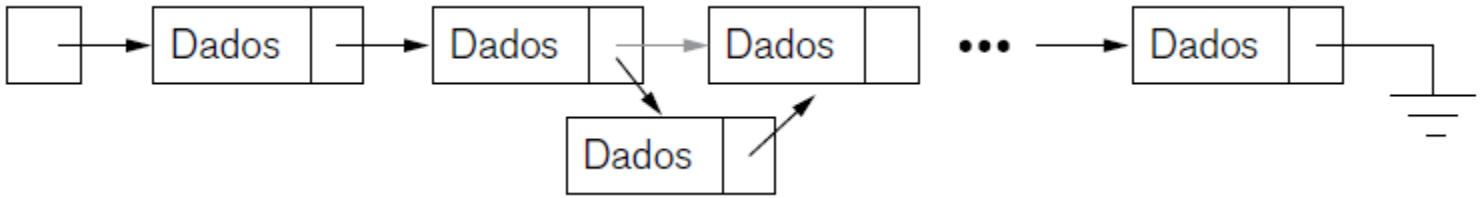
- A lista possui um nó especial, chamado nó-cabeça. O nó-cabeça indica o início da lista encadeada e nunca pode ser removido. Dados (registros) que são armazenados nos demais nós da lista, não devem ser armazenados no nó-cabeça.

Exemplo de inserção:

- A partir do nó-cabeça, efetua-se a busca da posição onde o novo nó deverá ser inserido.



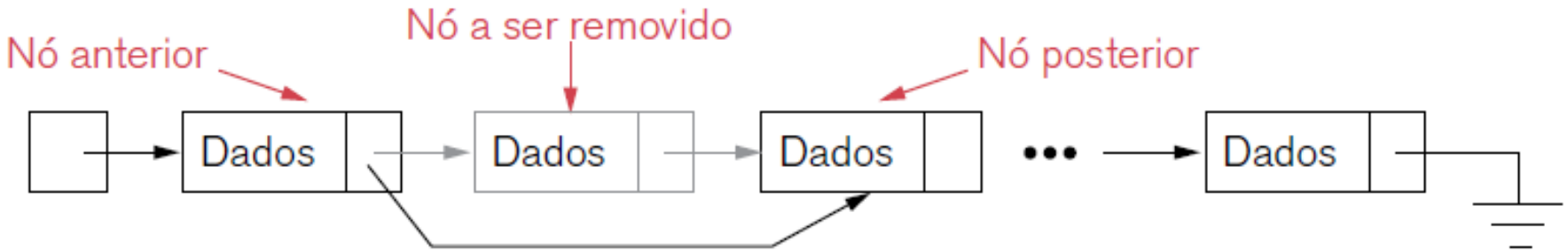
- Uma vez encontrado o local de inserção, liga-se o ponteiro do nó anterior ao novo nó e liga-se o ponteiro no novo nó ao nó posterior.



LISTA LINEAR ENCADEADA

Exemplo de remoção:

- A partir do nó-cabeça, efetua-se a busca do nó a ser removido.
- Uma vez encontrado nó a ser removido, liga-se o ponteiro do nó anterior ao nó posterior.



LISTA LINEAR ENCADEADA

Vantagens da lista linear encadeada:

- Facilidade de inserir ou remover um elemento em qualquer ponto da lista.
- Não há a necessidade de movimentar os elementos da lista quando há uma inserção ou remoção de elemento.

Desvantagem da lista linear encadeada :

- Por utilizar ponteiros, a implementação deve ser feita com cuidado para que não ocorra um mal encadeamento (ligação) dos nós e consequentemente a lista seja perdida.
- Encontrar um determinado elemento na posição n da lista é necessário percorrer os n-1 anteriores.
- Necessidade de memória extra para armazenamento dos ponteiros.

Exemplo

- A primeira contém os dados do nó: código do aluno, nome e turma.
- A segunda, contém o ponteiro, * ptrLink, que apontará para o próximo nó. Como o próximo nó da lista, será uma estrutura do tipo DADOS_ALUNO, o ponteiro foi declarado para permitir apontar também para uma estrutura do tipo DADOS_ALUNO.

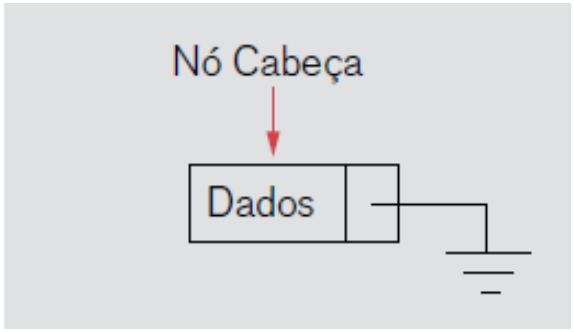
```
struct DADOS_ALUNO{
     int CodAluno;
     char Nome[100];
     int Turma;

     struct DADOS_ALUNO * ptrLink;
};
```

Exemplo

- O ponto inicial da lista é o nó cabeça e é alocada a memória para conter a estrutura DADOS_ALUNO, necessária ao ponteiro *ptrCabeca.
- Neste caso, os membros CodAluno, Nome e Turma não são utilizados. Utiliza-se apenas o membro *ptrLink (ponteiro que apontará para o próximo nó da lista).
- Como ainda não possui outros nós, o ponteiro ptrCabeca->ptrLink aponta para NULL (endereço vazio), para não apontar para um endereço inválido.

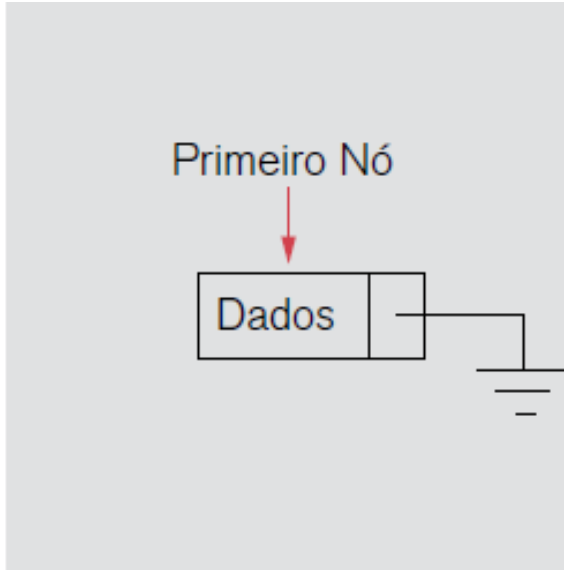
```
// Aloca memória para a estrutura
ptrCabeca = new DADOS_ALUNO;
// Aponta para um endereço vazio
ptrCabeca->ptrLink = NULL;
```



Exemplo

- Criado o primeiro nó, aloca-se o espaço de memória necessário.
- Atribuem-se valores as variáveis membros da estrutura.
- Como o primeiro nó não possui um nó a sua frente, o ponteiro ptrPrimeiraNo->ptrLink aponta para um endereço vazio.

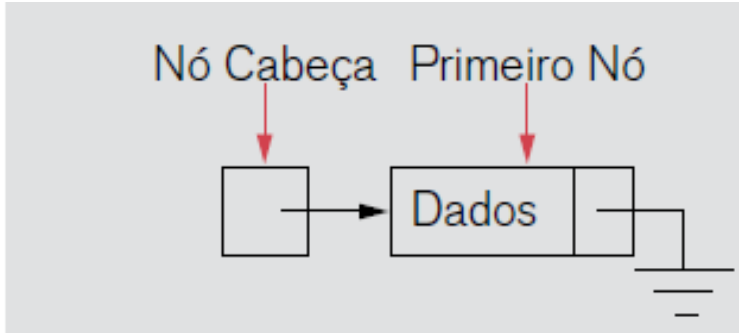
```
// Aloca memória para a estrutura
ptrPrimeiraNo = new DADOS_ALUNO;
// Atribui valores aos membros
ptrPrimeiraNo->CodAluno = 10;
strcpy(ptrPrimeiraNo->Nome, "José");
ptrPrimeiraNo->Turma = 250;
// Aponta para um endereço vazio
ptrPrimeiraNo->ptrLink = NULL;
```



Exemplo

- Até o momento foi feito apenas a criação e atribuição de valores ao nó ptrPrimeiraNo.
- O próximo passo é colocá-lo na lista, isto é feito ligando este nó ao último nó criado (nó cabeça).

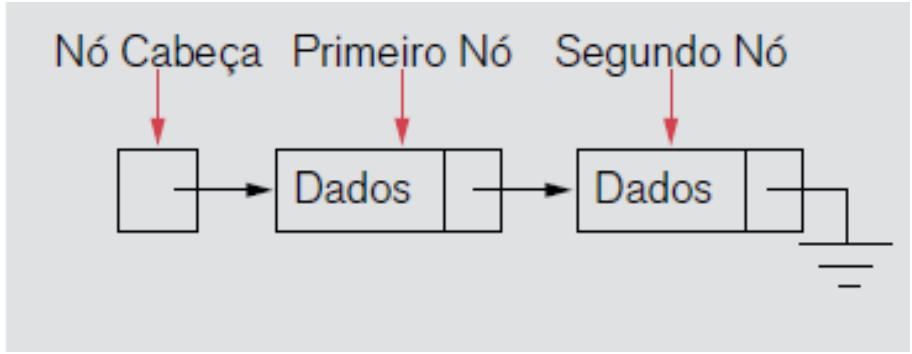
```
// Liga o primeiro nó ao nó cabeça
ptrCabeca->ptrLink = ptrPrimeiraNo;
```



Exemplo

- Atribui-se os valores para os membros do segundo nó.
- O próximo passo é colocá-lo na lista, isto é feito ligando este nó ao último nó criado (primeiro nó).

```
// Liga o segundo nó ao primeiro
nó
ptrPrimeiraNo->ptrLink =
ptrSegundoNo;
```

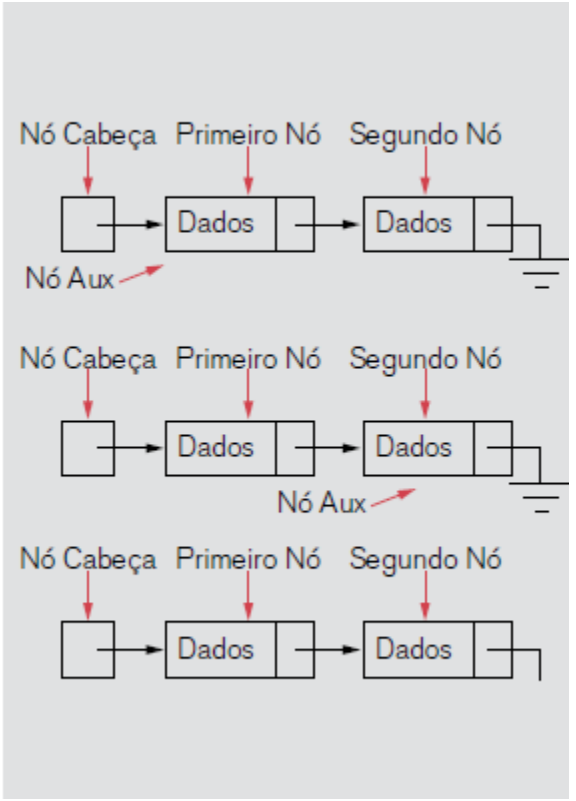


Exemplo

- Mostra os valores

```
ptrAux = ptrCabeca->ptrLink;
Ind = 1;
while(ptrAux != NULL){
  cout <<"Nó: " << Ind << endl;
  cout <<"Código do Aluno: "<<
        ptrAux->CodAluno << endl;
  cout << "Nome: " << ptrAux->Nome << endl;
  cout << "Turma: " << ptrAux->Turma << endl;
  cout << endl << endl;

  Ind++;
  ptrAux = ptrAux->ptrLink;
}
```



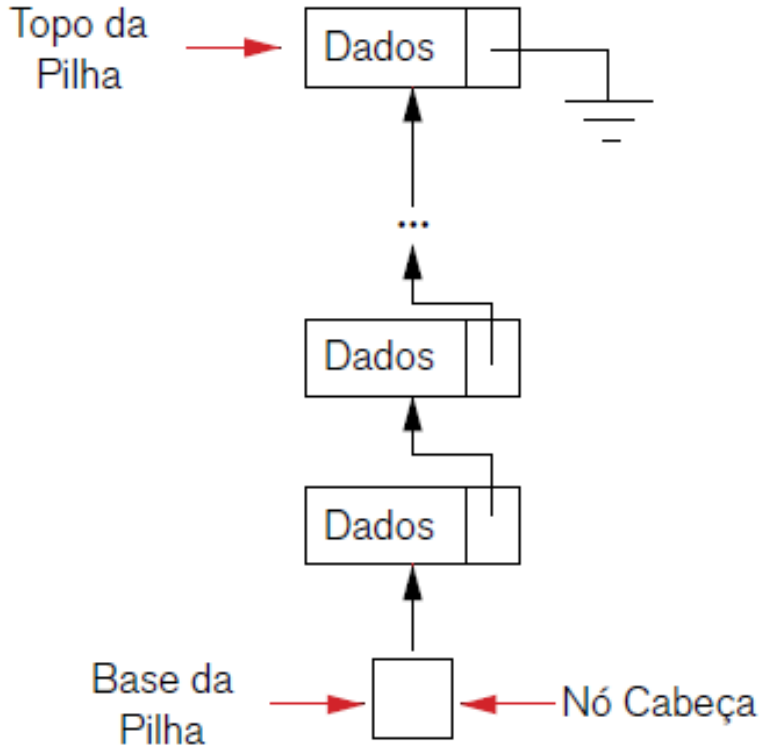
LISTA LINEAR ENCADEADA

Exemplo

- Criar uma lista vazia.
- Verificar lista vazia
- Inserir novo nó
- Localizar um nó
- Obter tamanho
- Exibir lista
- Remover nó

PILHA ENCADEADA

- A pilha simplesmente encadeada também é conhecida como pilha dinâmica. A pilha dinâmica é um tipo de lista simplesmente encadeada que tem como característica a inserção e remoção dos nós no topo da pilha (LIFO).

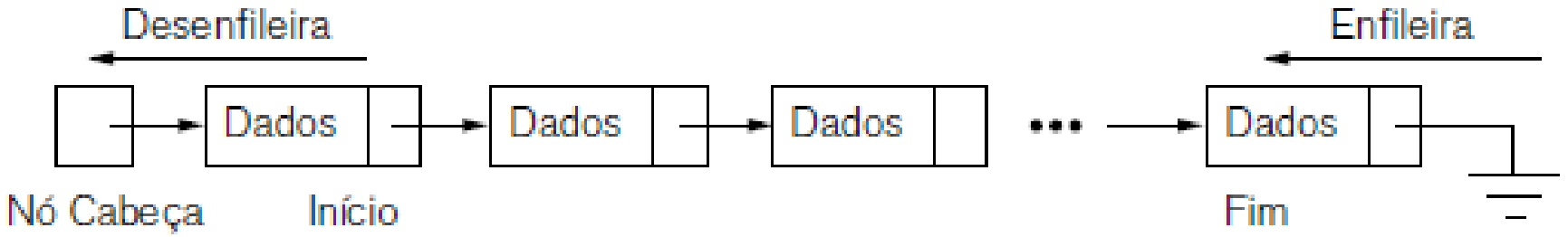


PILHA ENCADEADA

- Criar: cria uma pilha vazia.
- Verificar: verifica se pilha está vazia
- Empilhar: insere um novo elemento no topo da pilha.
- Desempilhar: remove um elemento do topo da pilha.
- Exibir topo: exibe o elemento do topo da pilha.
- Exibir a quantidade: retorna a quantidade de elementos da pilha.
- Esvaziar: esvazia todos os elementos da pilha.

FILA ENCADEADA

- A fila simplesmente encadeada, também conhecida como fila dinâmica. A fila dinâmica é um tipo de lista simplesmente encadeada que tem como característica a inserção dos nós inseridos em uma extremidade, e retirados na extremidade oposta (FIFO).

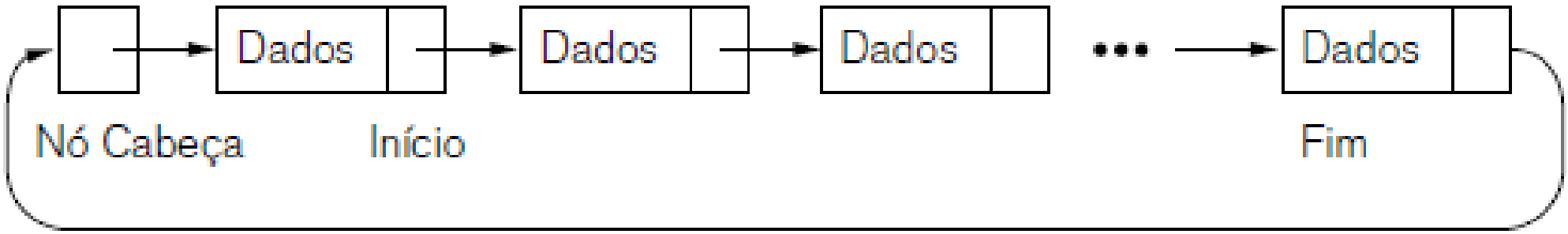


FILA ENCADEADA

- Criar: cria uma fila vazia.
- Verificar: verifica se fila está vazia
- Enfileirar: insere um elemento no fim da fila.
- Desenfileirar: remover um elemento no início da fila.
- Exibir início: exibe o elemento do início da fila.
- Exibir a quantidade: retorna a quantidade de elementos da fila.
- Esvaziar: esvazia a fila.

LISTA CIRCULAR ENCADEADA

- As listas circulares diferem das listas encadeadas simples com relação ao seu último nó. Enquanto uma lista encadeada simples tem o seu último nó apontando para um endereço vazio, NULL, uma lista circular aponta para o primeiro. Isto permite que se possa percorrer todos os nós da lista sem acessar uma posição inválida, ou seja, o NULL.

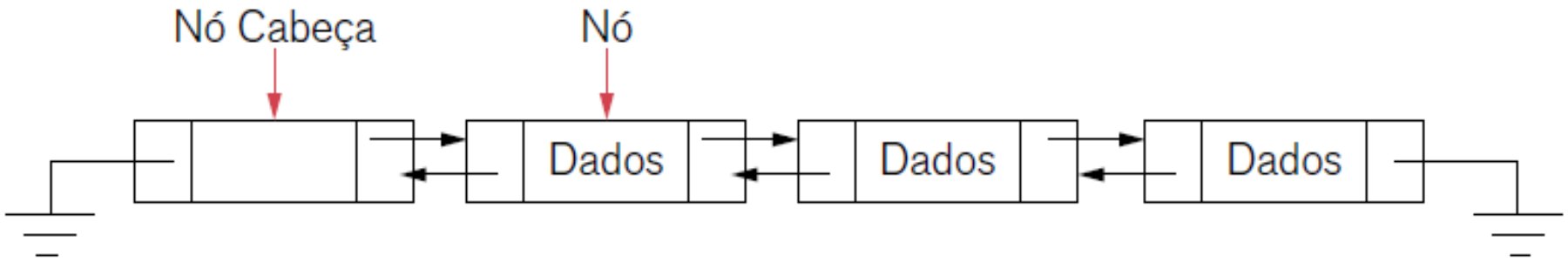


LISTA CIRCULAR ENCADEADA

- Criar: cria uma lista vazia.
- Verificar: verifica se lista está vazia
- Inserir: insere um novo nó no fim da lista.
- Remover: remove um nó da lista.
- Exibir: exibe a lista

LISTA DUPLAMENTE ENCADEADA

- Uma deficiência encontrada nas listas circulares simplesmente encadeadas é não poder percorrer os nós em ordem inversa, ou seja, do final para o início.
- A solução para este tipo de situação são as listas duplamente encadeadas. A estrutura de uma lista duplamente encadeada mantém dois links: um para o próximo nó e um para o nó anterior. Este arranjo permite percorrer a lista em ambas as direções.



LISTA DUPLAMENTE ENCADEADA

- Criar: cria uma lista vazia.
- Verificar: verifica se lista está vazia
- Localizar: localiza um nó
- Inserir: insere um novo nó no fim da lista.
- Remover: remove um nó da lista.
- Exibir: exibe a lista