Listas - Tipo Abstrato de Dados - ADT

Listas é outro tipo abstrato de dados (ADT), ou seja, é uma especificação de um conjunto de dados.

Uma lista consiste de um conjunto de dados organizados em ordem linear.

Os algoritmos que manipulam uma estrutura de dados do tipo listas fazem com que o conjunto de dados cresçam, encolham ou sofram alterações ao longo do tempo.

ADT - Listas

Uma lista é uma estrutura de dados que representa uma coleção de itens sequenciais.

Uma lista armazena elementos dispostos um após o outro, como em uma lista de nomes, dados de um funcionário, de um produto, etc.

Uma lista pode ser ordenada ou não.

ADT - Listas

A implementação pode ocorrer de duas maneiras:

- **Vetores** (lista estática, porque tem-se o uso de endereços contíguos de memória e a ordem linear é determinada pelos índices do vetor);
- **Lista encadeada** (lista dinâmica, quando além do dado, possui também um ponteiro para o próximo elemento).

Quando uma lista contém apenas um dado primitivo, como um número, chama-se **lista homogênea**, já quando um elemento de uma lista contém um dado composto, como o nome e o salário de um funcionário, chama-se **lista heterogênea**.

ADT – Listas – Tipos

A estrutura lista pode ser representada em cinco tipos diferentes, e cada um pode ser implementado de forma estática ou dinâmica:

- **Lista simplesmente encadeada e não ordenada;**
- **Lista simplesmente encadeada e ordenada;**
- **Lista duplamente encadeada e não ordenada;**
- **Lista duplamente encadeada e ordenada;**
- **Listas circulares.**

ADT - Listas

Implementação por vetores (características):

- Acesso rápido aos dados;
- Tamanho máximo da lista pré-determinado;
- Necessidade de mover elementos durante a inserção e remoção de elementos na lista.

Implementação por lista encadeada (características):

- Tamanho da lista é dinâmico;
- Código reaproveitável das outras estruturas de dados;
- Acesso lento a um elemento específico;
- Inserção e remoção rápida de elementos na lista.

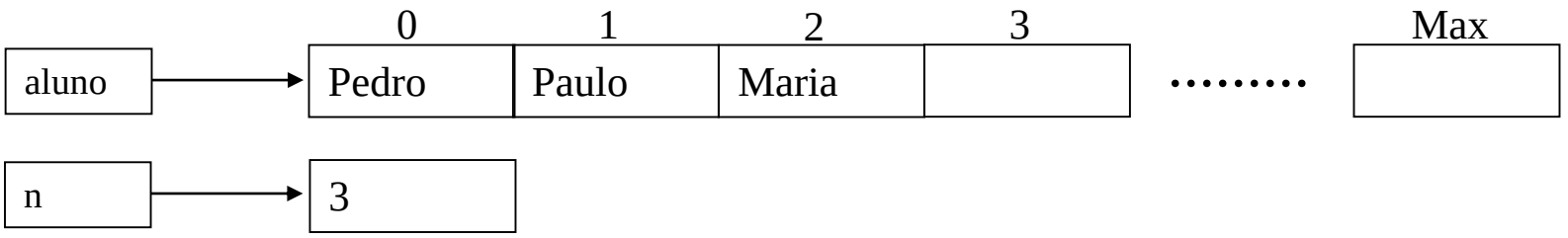
ADT – Listas – Quando usar o quê

A lista encadeada deve ser utilizada **sempre que a quantidade de dados a ser armazenada não puder ser prevista ou quando os dados forem inseridos ou eliminados com frequência**.

A lista encadeada obtém qualquer espaço necessário quando novos itens são adicionados, portanto, pode-se expandir para preencher toda a memória disponível; e não há necessidade de preencher "buracos" durante a eliminação, como há em vetores.

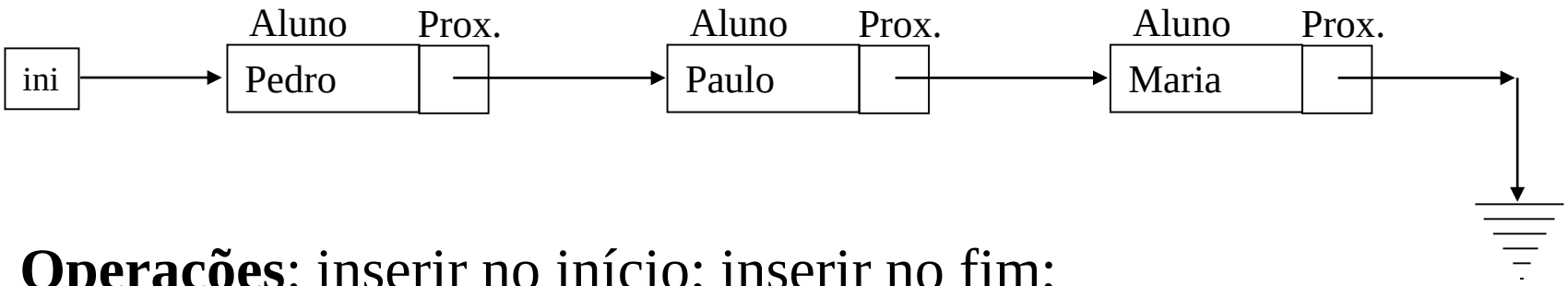
ADT - Lista por Vetores

Uma lista implementada por vetores é formada por uma sequência de elementos instanciados de forma contígua na memória do computador.
De tal forma, que os possam ser encontrados facilmente, bastando saber qual a posição do primeiro elemento da lista.



ADT - Lista Encadeada

Uma lista simplesmente encadeada não ordenada é formada por uma sequência de nós. Cada nó contém a informação relevante e uma referência indicando onde se encontra o próximo nó da lista.



Operações: inserir no início; inserir no fim;
consultar toda a lista; remover um elemento qualquer; esvaziá-la.

Ordenamento de uma Lista

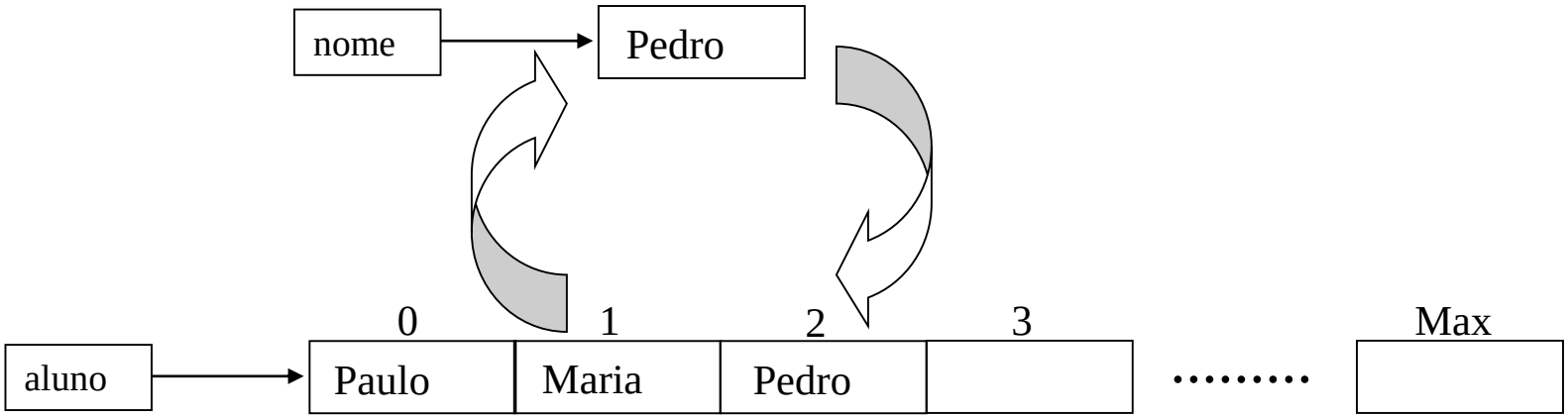
Para certas aplicações será útil manter os dados em ordem classificada na lista. Uma lista com essa característica é chamada de lista ordenada.

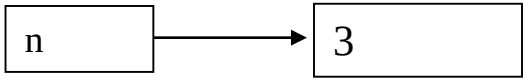
Em uma lista ordenada, os itens são organizados na ordem especificada. A eliminação está geralmente limitada ao menor item (ou maior) na lista, que está no início da lista, embora algumas vezes métodos *find*() e *delete*(), que buscam a lista para obter nós específicos, sejam usados também.

Ordenamento de uma Lista

Não há regras quanto à inserção direta numa lista ordenada (objeto *SortedList* do Java) ou não. Todavia pode-se executar operações manuais de ordenação antes de um processo específico, como por exemplo, listar a lista (ordenada); simulando assim uma lista ordenada.

Ordenamento de uma Lista

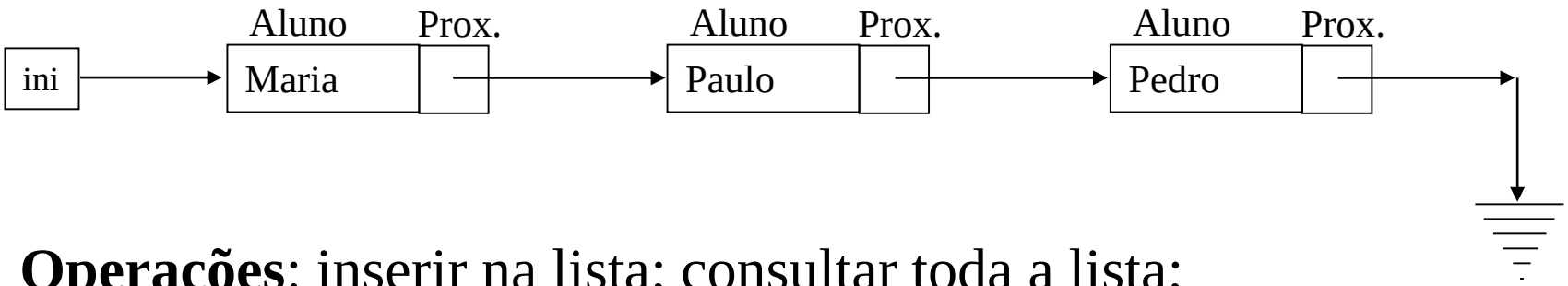




Método bolha, garante que o maior estará no final ao término do primeiro ciclo

ADT - Lista Encadeada

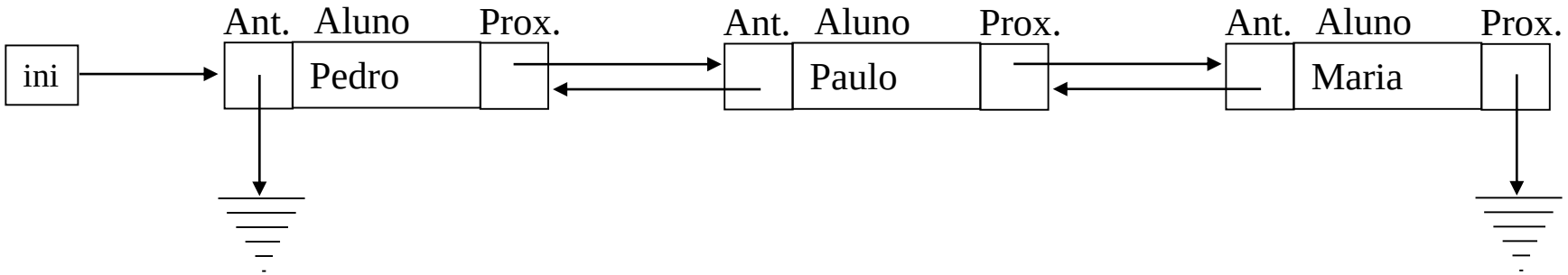
Uma lista simplesmente encadeada ordenada é formada por uma sequência de nós. Cada nó contém a informação relevante e uma referência indicando onde se encontra o próximo nó da lista.



Operações: inserir na lista; consultar toda a lista; remover um elemento qualquer; esvaziá-la.

ADT - Lista Encadeada

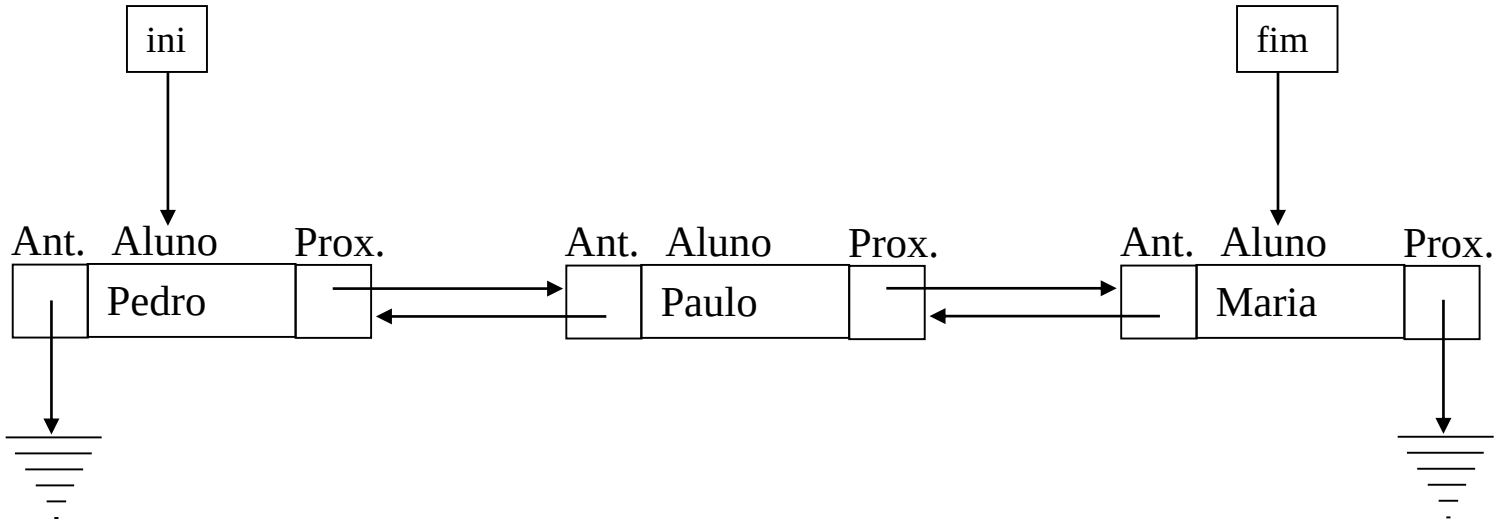
Uma lista duplamente encadeada e não ordenada é formada por uma sequência de nós. Cada nó contém a informação relevante e duas referências. A primeira indica onde se encontra o nó anterior da lista, e a segunda indica onde se encontra o próximo nó da lista.



Operações: inserir no início; inserir no fim;
consultar toda a lista do início ao fim ou do fim ao início;
remover um elemento qualquer; esvaziá-la.

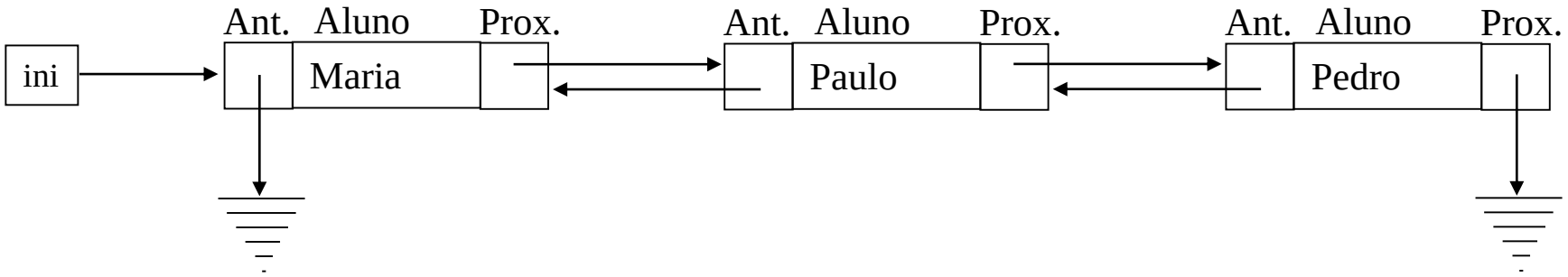
ADT - Lista Encadeada

Os ponteiros (ou referências) que indicam o início e o fim numa lista encadeada são chamados de sentinela ou lista com extremidade dupla.



ADT - Lista Encadeada

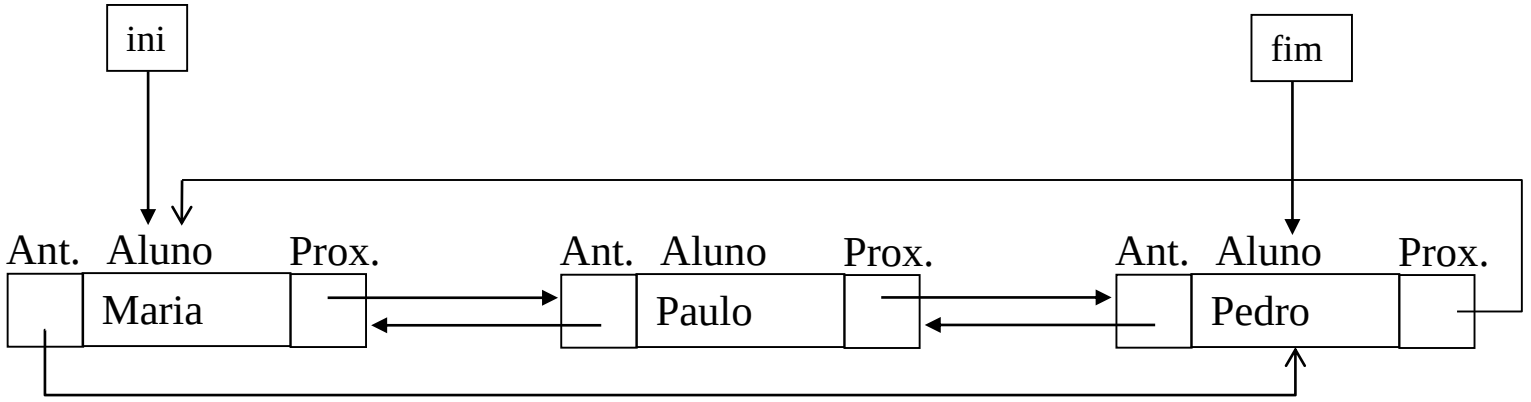
Uma lista duplamente encadeada e ordenada é formada por uma sequência de nós. Cada nó contém a informação relevante e duas referências. A primeira indica onde se encontra o nó anterior da lista, e a segunda indica onde se encontra o próximo nó da lista.



Operações: inserir na lista; consultar toda a lista do início ao fim ou do fim ao início; remover um elemento qualquer; esvaziá-la.

ADT – Listas Circulares

No caso de uma lista duplamente encadeada circular e ordenada (as características da circular, equivalem as vistas para a estrutura de dado fila), a referência final terá como referência o primeiro nó. E no primeiro nó, a referência do anterior receberá o último nó da lista.



Também pode-se implementar listas circulares para listas simplesmente encadeadas, sejam ordenadas ou não.

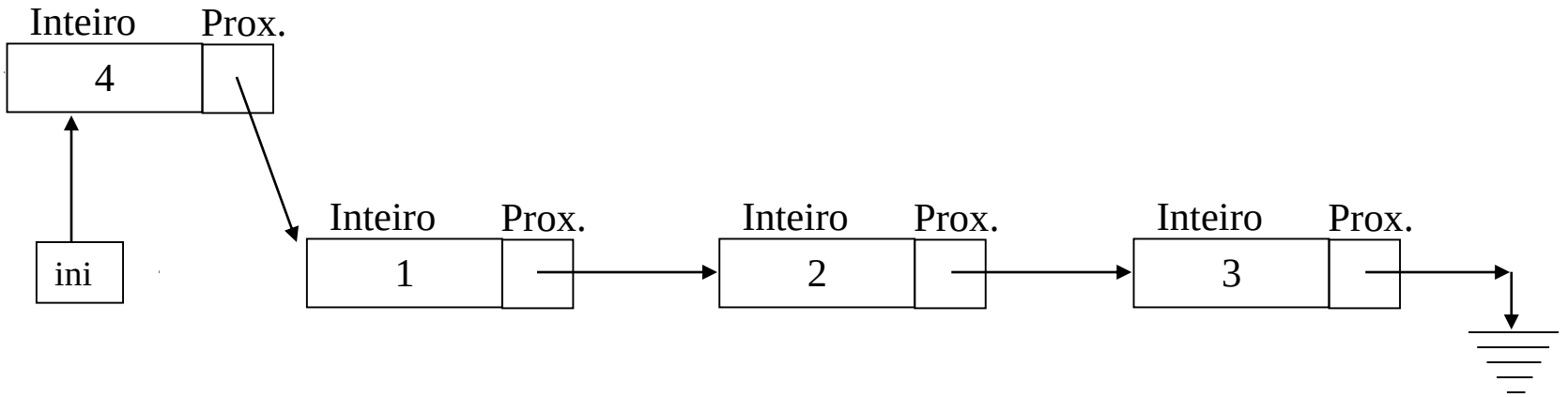
Análise: INSERÇÃO NO INÍCIO DA LISTA

É necessário apenas realizar operações de atribuições atualizando a referência do início (e fim da lista, quando necessário).

Para implementação nas listas:

- Lista simplesmente encadeada e não ordenada;
- Lista duplamente encadeada e não ordenada;
- Lista circular simplesmente encadeada e não ordenada;
- Lista circular duplamente encadeada e não ordenada.

INSERÇÃO NO INÍCIO DA LISTA



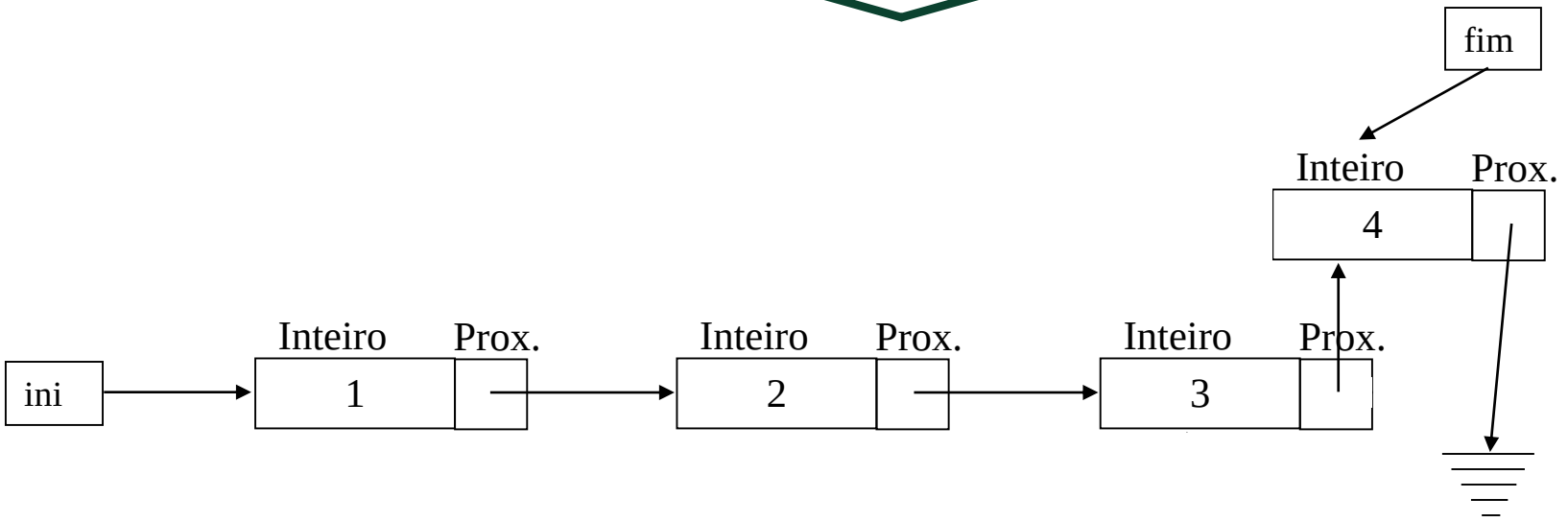
Análise: INSERÇÃO NO FIM DA LISTA

É necessário apenas realizar operações de atribuições atualizando a referência do fim (e início da lista, quando necessário).

Para implementação nas listas:

- Lista simplesmente encadeada e não ordenada;
- Lista duplamente encadeada e não ordenada;
- Lista circular simplesmente encadeada e não ordenada;
- Lista circular duplamente encadeada e não ordenada.

INSERÇÃO NO FIM DA LISTA



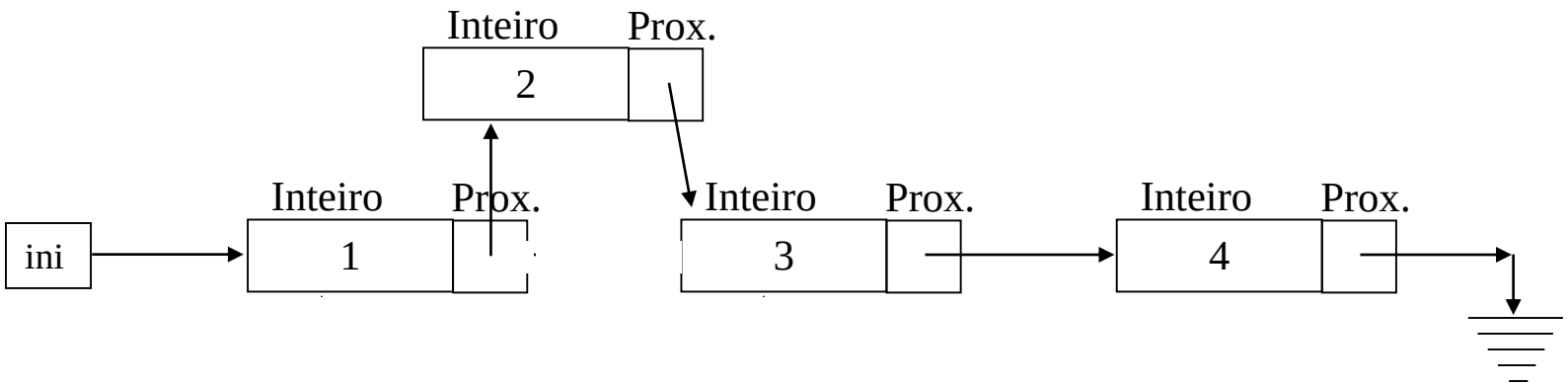
Análise: INSERÇÃO ORDENADA

O elemento a ser inserido pode ser, no pior caso, o maior de todos os já existentes na lista. Com isso, percorre-se toda a lista, comparando o novo elemento com os demais, para encontrar a posição correta e inseri-lo.

Para implementação nas listas:

- Lista simplesmente encadeada e ordenada;
- Lista duplamente encadeada e ordenada;
- Lista circular simplesmente encadeada e ordenada;
- Lista circular duplamente encadeada e ordenada.

INSERÇÃO ORDENADA

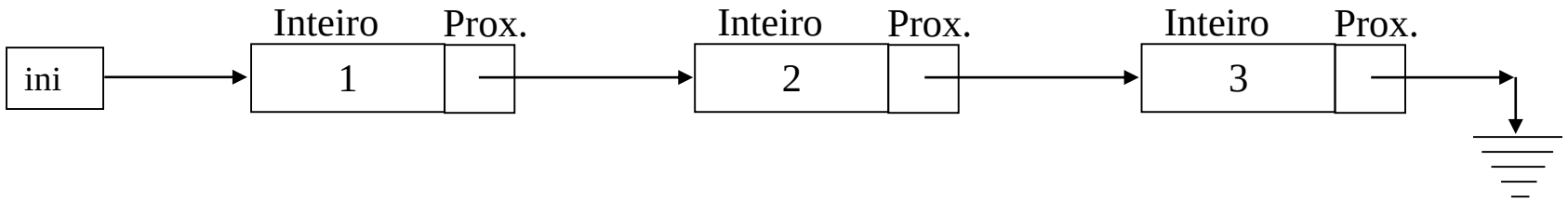


Análise: CONSULTAR TODA A LISTA

Em todos os tipos de listas, para realizar a operação de consultar toda a lista é necessário acessar cada elemento.

No caso da lista duplamente encadeada, ordenada ou não, é possível percorrer a lista do início ao fim ou vice-versa.

CONSULTAR TODA A LISTA

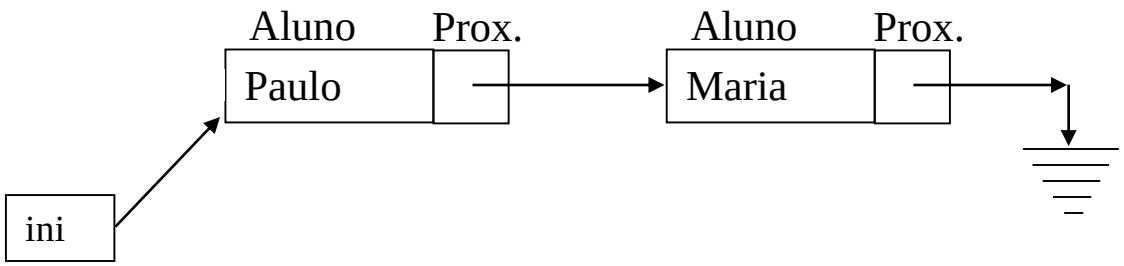


Análise: REMOÇÃO

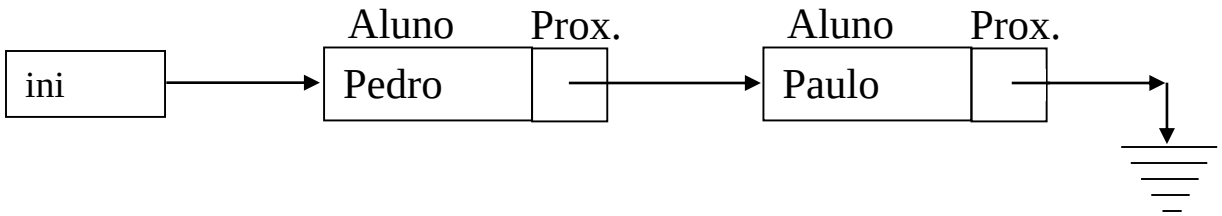
A remoção de qualquer tipo de lista, consiste em procurá-lo e depois acertar as referências para que a lista continue sendo acessada após a remoção.

Na busca pelo elemento a ser removido, percorre-se, no pior caso, todos os elementos da lista.

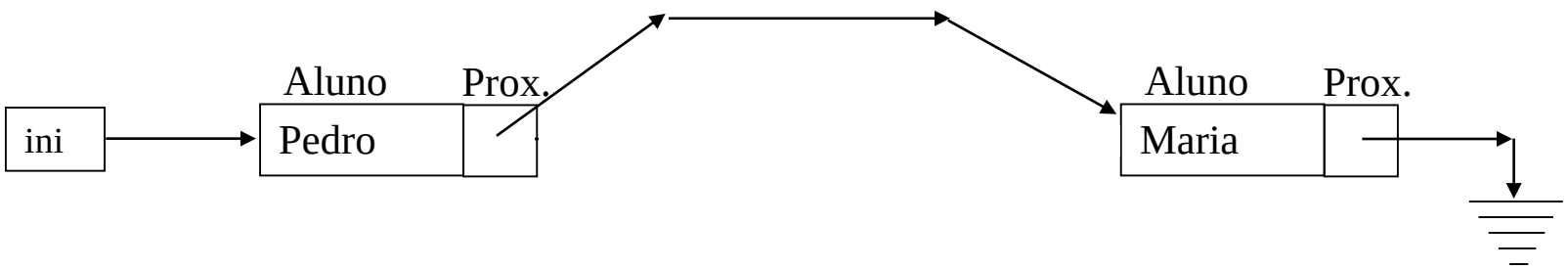
REMOÇÃO NO INÍCIO



REMOÇÃO NO FIM



REMOÇÃO NO MEIO



Análise: ESVAZIAR A LISTA

A operação de esvaziamento da lista consiste em remover todos os elementos dela.

O tempo gasto nessa operação depende da linguagem de programação que está sendo utilizada.

No caso das linguagens JAVA e PYTHON, não é necessário realizar a remoção de cada um dos elementos, apenas atualizar a referência para o início (e/ou fim) da lista em nulo e as memórias alocadas serão desalocadas por um procedimento de coleta de lixo das linguagens.

Exercícios - Lista por Vetor

Sugere-se que, a cada manipulação da lista, a mesma seja impressa em sua totalidade.

01) Construa um programa para incluir uma lista de nomes (inclusão no final da lista).

02) Altere o programa anterior, para permitir a inclusão de um nome no início da lista.

03) Altere o programa anterior, para permitir a inclusão de um nome no meio da lista, ou após uma “chave”.

04) Altere o programa anterior, para permitir a exclusão do primeiro nome.

05) Altere o programa anterior, para permitir a exclusão do último nome.

06) Altere o programa anterior, para permitir a exclusão de um nome no meio da lista, ou de uma “chave”.

07) Altere o programa anterior, para permitir a exclusão de todos os nomes da lista.

08) Refaça o exercício 07, mas agora, que permita manipular inteiros.

Exercício 09

Construa um programa para manipular inteiros em uma lista simplesmente encadeada e não ordenada:

1 – Inserir no final da lista
2 – Inserir no início da lista
3 – Excluir um elemento
4 – Consultar a lista
5 – Esvaziar a lista

Exercício 10

Construa um programa para manipular inteiros em uma lista simplesmente encadeada e ordenada:

1 – Inserir um inteiro na lista
2 – Excluir um inteiro na lista
3 – Consultar a lista
4 – Esvaziar a lista

Exercício 11

Construa um programa para manipular inteiros em uma lista duplamente encadeada e não ordenada:

1 – Inserir no final da lista
2 – Inserir no início da lista
3 – Excluir um elemento
4 – Consultar a lista
5 – Esvaziar a lista

Exercício 12

Construa um programa para manipular inteiros em uma lista duplamente encadeada e ordenada:

1 – Inserir um inteiro na lista
2 – Excluir um inteiro na lista
3 – Consultar a lista
4 – Esvaziar a lista

Exercício 13

Refaça os exercícios 09, 10, 11 e 12, mas agora para listas circulares.

Exercício 14 – BÔNUS

Escreva um programa para simular um sistema de computadores multiusuários, como segue: cada usuário tem um ID exclusivo e deseja executar algumas operações no computador. Entretanto, somente uma transação pode ser processada pelo computador em determinado momento. Cada linha de entrada representa um único usuário e contém o ID do usuário seguido de uma hora de início e de uma sequência de inteiros representando a duração de cada uma de suas transações. A entrada é classificada pela hora crescente de início, e todas as marcações de hora e de duração são em segundos.

Presuma que um usuário não solicitará tempo para uma transação até que a transação anterior tenha terminado e que o computador aceite transações baseado na ordem de chegada (primeiro a chegar, primeiro a ser atendido).

O programa deve simular o sistema e imprimir uma mensagem contendo o ID do usuário e a hora sempre que uma transação começar e terminar.

No final da simulação, ele deve imprimir o tempo médio de espera para uma transação. (O tempo de espera é o intervalo de tempo entre a hora em que a transação foi solicitada e a hora em que ela foi iniciada.)

Exercício 15 – BÔNUS

Examine a seguinte variação do problema de Josephus. Um grupo de pessoas faz um círculo e cada uma escolhe um inteiro positivo. Um de seus nomes e um inteiro positivo n são escolhidos. Começando com a pessoa cujo nome é escolhido, elas serão contadas ao redor do círculo, no sentido horário, e a enésima pessoa será eliminada. O inteiro positivo que essa pessoa escolheu será, então, usado para continuar a contagem. Toda vez que uma pessoa for eliminada, o número que ela escolheu será usado para determinar a próxima pessoa eliminada. Por exemplo, suponha que as cinco pessoas sejam A, B, C, D e E, que elas tenham escolhido os números 3, 4, 6, 2 e 7, respectivamente, e que o inteiro 2 seja inicialmente escolhido. Sendo assim, se começarmos a partir de A, a sequência na qual as pessoas serão eliminadas do círculo será B, A, E, C, deixando D como o último no círculo.
Escreva um programa que leia um grupo de linhas de entrada. Cada linha de entrada, exceto a primeira e a última, contém um nome e um inteiro positivo escolhidos por essa pessoa. A ordem dos nomes nos dados é a sequência em sentido horário das pessoas no círculo, e a contagem deve começar com o primeiro nome na entrada. A primeira linha de entrada contém o número de pessoas no círculo. A última linha de entrada contém somente um único inteiro positivo representando a contagem inicial. O programa imprime a sequência na qual as pessoas são eliminadas do círculo.

Exercício 16 – BÔNUS – Lista dupla circular

Escreva um programa com um menu de opções para executar cada uma das seguintes operações (considere elementos do tipo inteiro):

a. Incluir um elemento no final de uma lista (opção para lista A e para lista B);

b. Concatenar duas listas (lista A e B em C);

c. Liberar todos os nós numa lista (lista A ou B);

d. Inverter uma lista de modo que o último elemento se torne o primeiro, e assim por diante (lista A);

e. Eliminar o último elemento de uma lista (lista A);

f. Eliminar o segundo elemento de uma lista (lista A);

g. Combinar duas listas (A e B) numa única lista em ordem ascendente (lista C);

h. Inserir um elemento depois do terceiro elemento de uma lista (lista A);

i. Retornar a soma dos inteiros numa lista (lista A e B);

j. Retornar o número de elementos numa lista (lista A e B);

k. Criar uma segunda cópia de uma lista (lista A em C).