Estrutura de Dados

Aleardo Manacero Jr.

Programa

- Tipos abstratos de Dados (semana 1)
- Listas, pilhas e filas (semana 1)
- Árvores (semanas 2, 3 e 4)
- Tabelas Hash (espalhamento) (semana 5)
- Gerenciamento de memória (semana 6)
- Grafos (semanas 7, 8 e 9)
- E.D. espaciais (semanas 10, 11 e 12)

Tipos Abstratos de Dados

Tipos Abstratos de Dados

- Conceito fundamental dentro da computação;
- Entre outras coisas é a base de orientação a objetos;
- A idéia é estabelecer associações implícitas entre dados (simples ou complexos) e os operadores que permitam a sua manipulação;
- Essa associação é “natural” para tipos simples

Tipos Abstratos de Dados

- Para tipos complexos um TAD define formalmente a estrutura dos dados e de seus operadores;

- Não são definidas as implementações dos operadores, mas apenas suas especificações;

- Linguagens orientadas a objetos permitem a implementação de TADs através do conceito de encapsulamento dos dados (objetos) e operadores (métodos).

Tipos Abstratos de Dados

- Por exemplo, pode-se definir um TAD fila.

- Nele os operadores seriam inserção e remoção da fila, ocorrendo em seu final e início, respectivamente

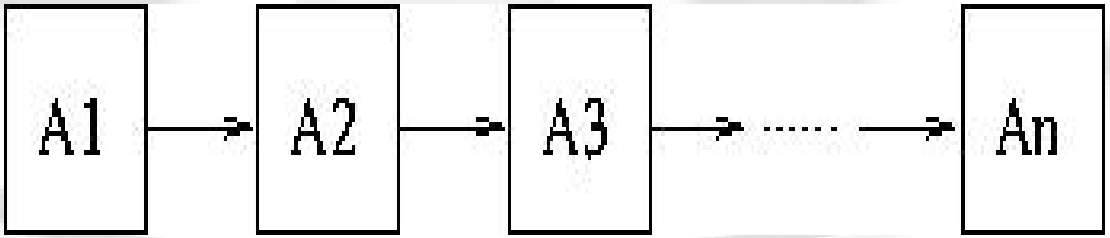
- Os dados seriam elementos da fila

- A implementação é dependente das estruturas disponíveis na linguagem utilizada e de opções de modelagem (estruturas estáticas ou dinâmicas)

Listas, pilhas e filas

Listas

- Listas são TADs que estruturam conjunto de dados para permitir o acesso a elementos individuais do conjunto de forma organizada e, possivelmente, eficiente
- Formalmente, uma lista é uma seqüência de elementos (A_1, A_2, até A_n), com posições específicas (1, 2, ..., n), em que A_i precede A_{i+1}, exceto para listas com menos de dois elementos



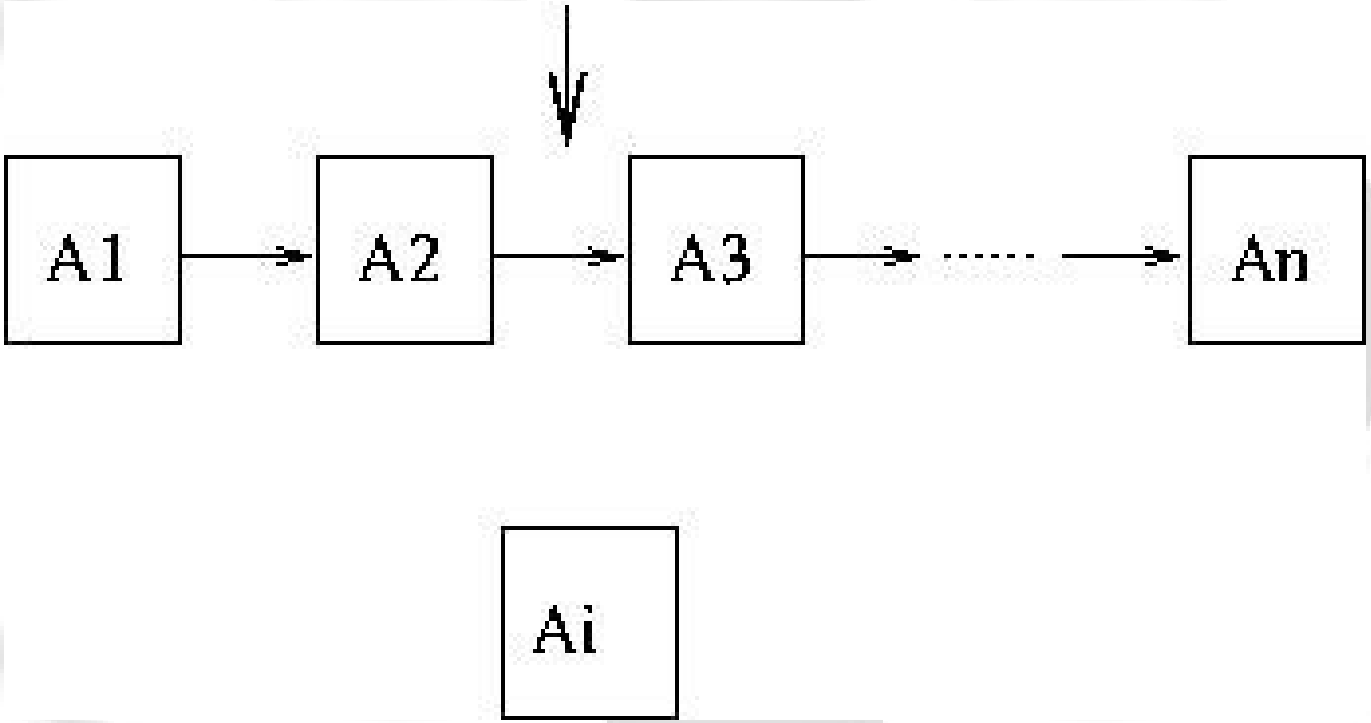
Listas

- O conjunto de operadores definidos para listas incluem:
  - Inserção
  - Remoção
  - Busca por elemento
  - Busca por posição
  - Impressão
  - Esvaziamento

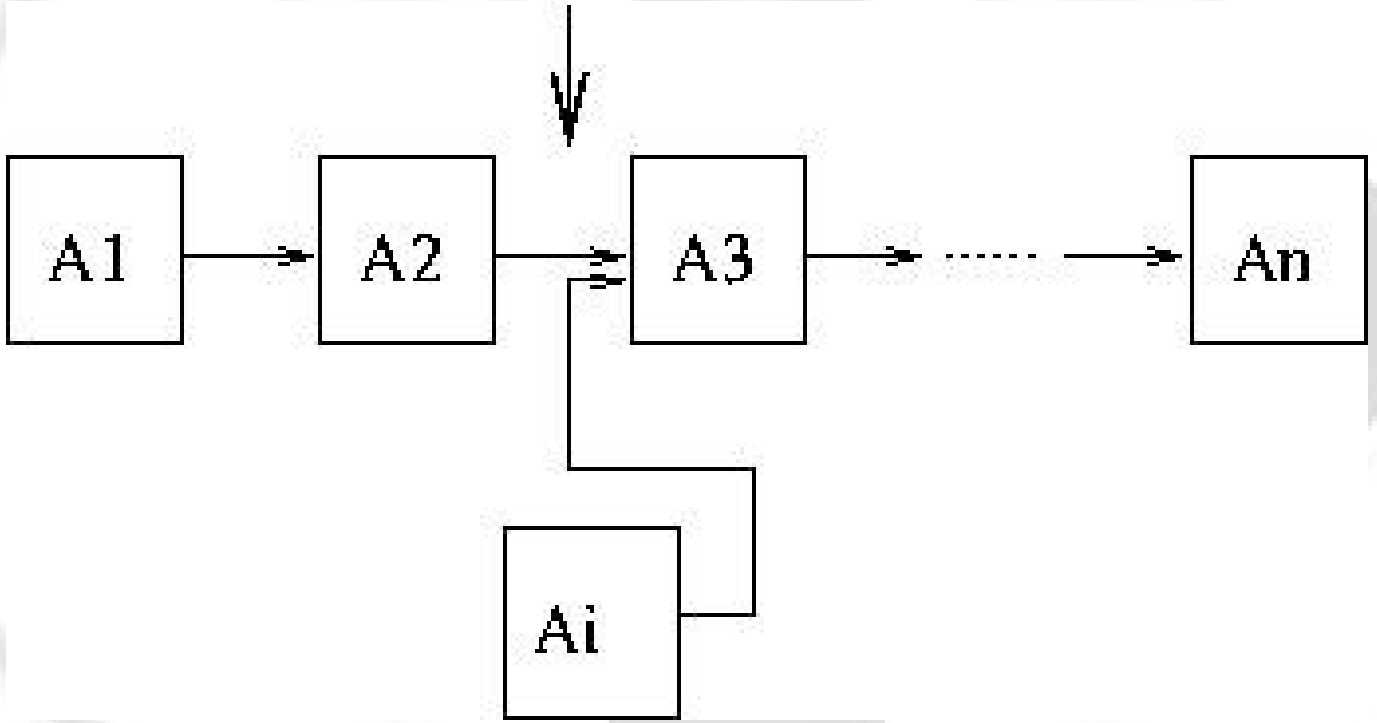
Listas - Inserção

- A operação de inserção deve considerar a posição em que o elemento será inserido na lista. Temos que considerar quatro casos:
  - Inserção numa lista vazia
  - Inserção no início de uma lista não vazia
  - Inserção no final de uma lista não vazia
  - Inserção no meio de uma lista não vazia

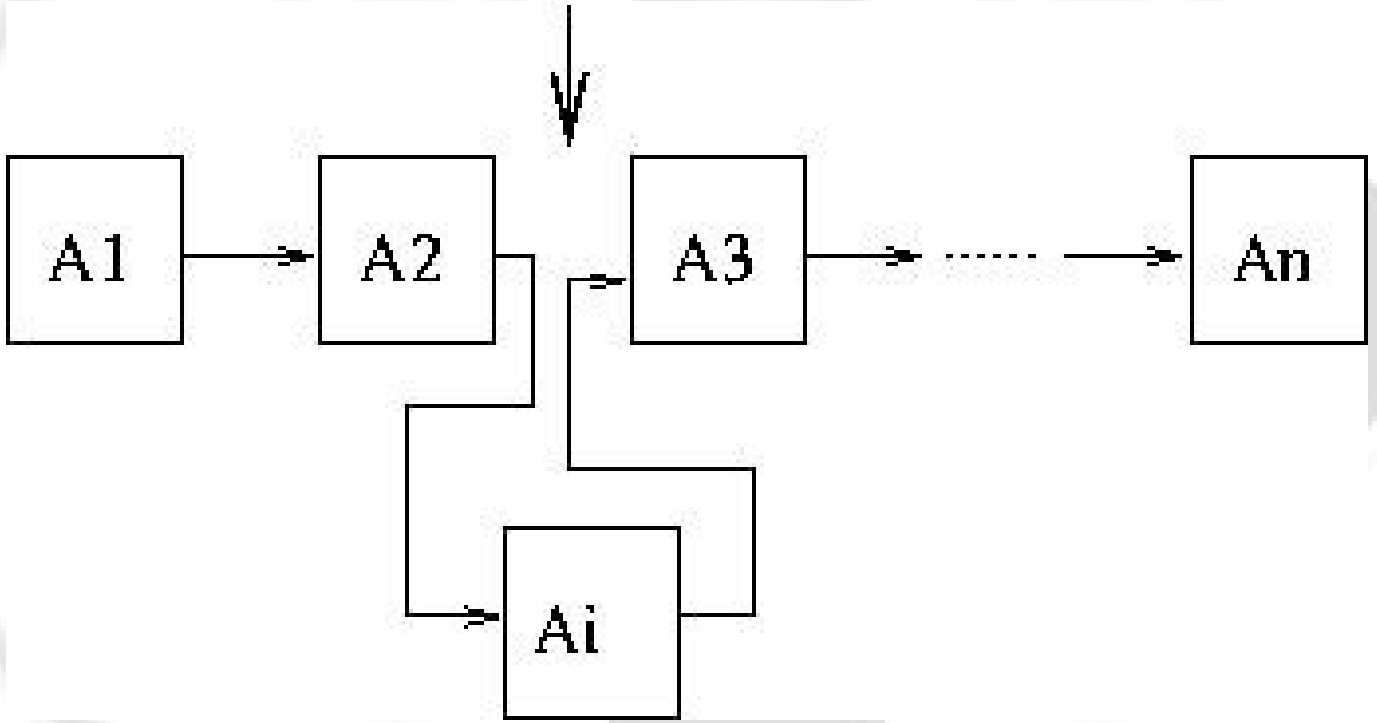
Inserção – Localizando a posição



Inserção – Reorganizando a lista (1)



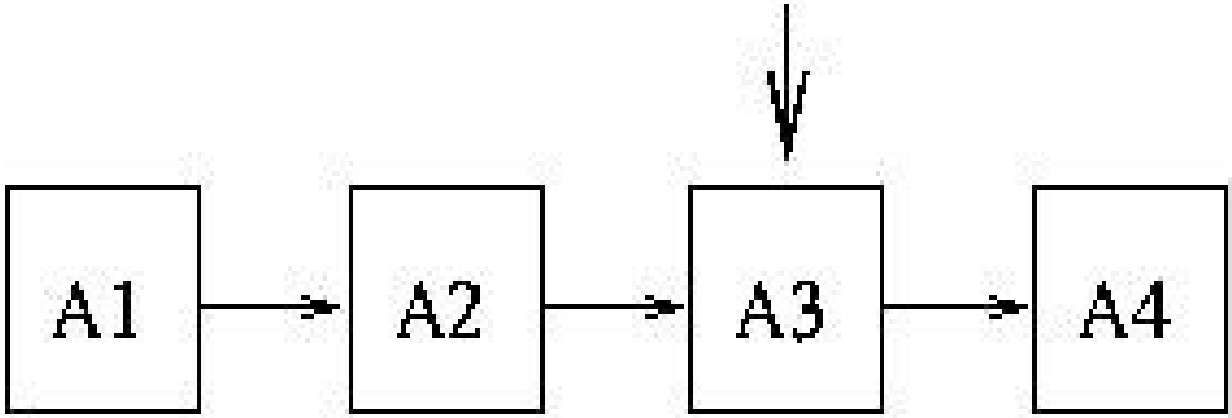
Inserção – Reorganizando a lista (2)



Listas - Remoção

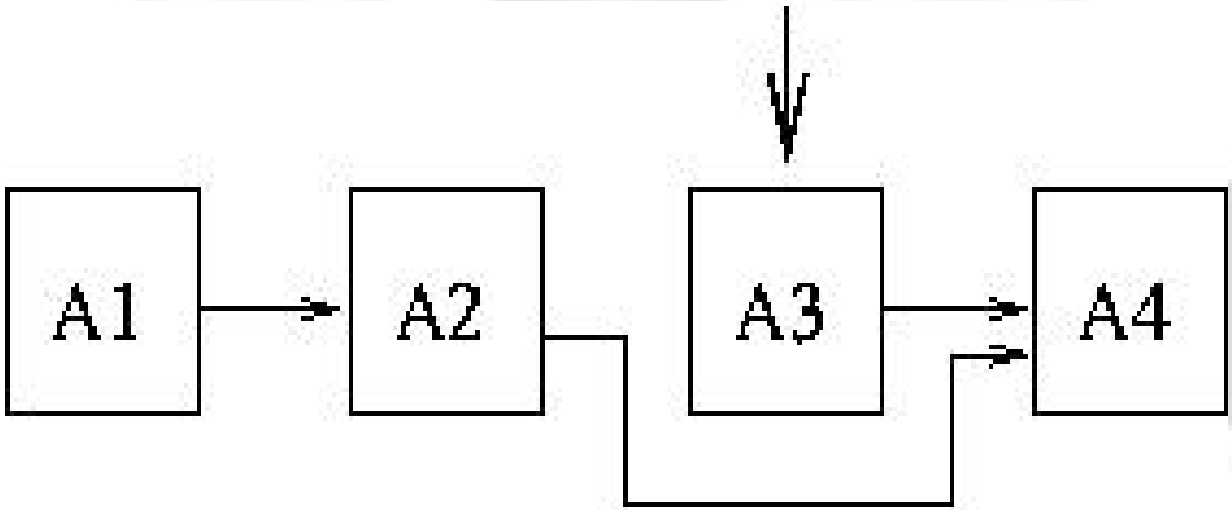
- Para a remoção de um elemento da lista devemos considerar três casos:
  - Remoção do último elemento da lista
  - Remoção do primeiro elemento da lista
  - Remoção de um elemento intermediário da lista

Remoção – Localizando a posição



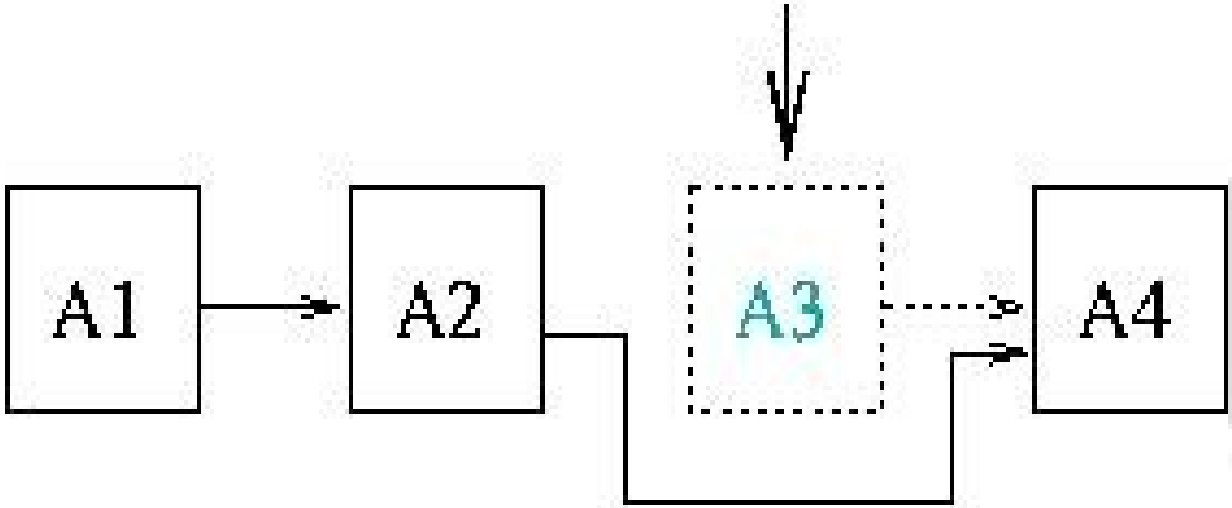
-

Remoção – Reorganizando a lista (1)



-

Remoção – Reorganizando a lista (2)



-

Listas - Buscas

- A busca de um elemento numa lista depende de características usadas na construção da lista

- Com isso veremos as condições de busca após verificarmos como são definidos os tipos especiais de listas (as listas ordenadas)

Listas ordenadas

- São listas em que a posição relativa de seus elementos deve considerar algum critério de precedência entre eles

- Por exemplo, se a lista contiver inteiros, uma ordenação crescente implicaria em que o elemento A_i será sempre menor ou igual ao elemento A_{i+1}

Inserção em listas ordenadas

- Aqui a operação de inserção implica, sempre, numa operação de busca pela posição de inserção

- Uma vez definida a posição em que o elemento deve ser inserido, o resto da operação segue os procedimentos descritos para uma lista simples

Remoção em listas ordenadas

- Aqui também a operação de remoção deve seguir os procedimentos adotados para uma lista simples

- A diferença fica na forma de se localizar, dentro da lista, a posição do elemento a ser retirado

- Isto também implica no uso de algum método de busca

Busca em listas simples

- A busca por um elemento em uma lista simples deve ser feita de forma exaustiva (busca linear sequencial)

- Isso implica em iniciar a busca a partir da primeira posição da lista e continuar até que se encontre o elemento ou se atinja o final da lista (nesse caso a busca seria mal sucedida)

Busca em listas ordenadas

- Quando temos listas ordenadas podemos fazer uso dessa informação e realizar a operação de busca de forma mais eficiente

- Um método possível é o da busca binária, em que a busca parte do elemento mais central da lista e prossegue considerando sub-listas cada vez menores a partir da posição relativa entre o elemento examinado e o elemento buscado

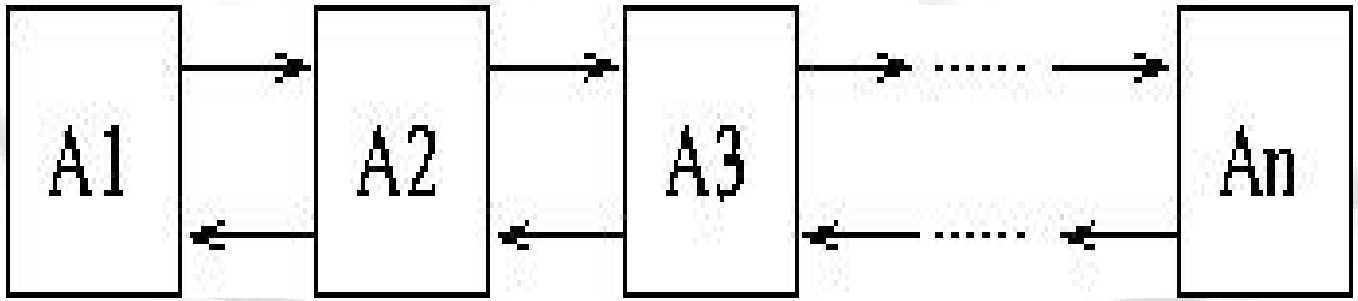
Busca em listas ordenadas

- Outro método é o da busca por interpolação

- Aqui usa-se o princípio da busca binária, porém dividindo-se as posições da lista de acordo com a proximidade relativa do valor buscado. Por exemplo, numa lista de 200 nomes ordenados alfabeticamente, é mais provável que Tadeu esteja próximo do final da lista e não de sua metade.

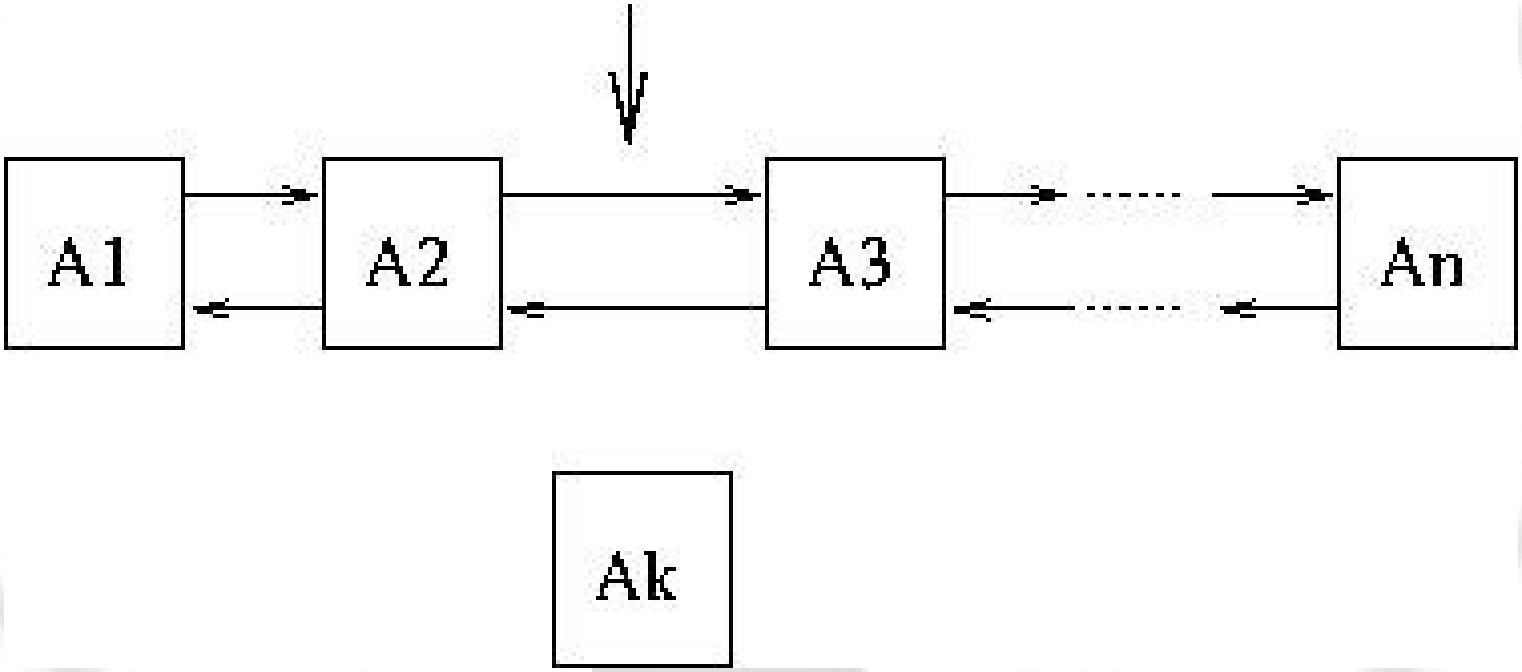
Listas duplamente encadeadas

- Outro tipo especial de lista é formado pelas listas duplamente encadeadas
- Nelas a ligação entre dois elementos consecutivos da lista é bidirecional, ou seja, tanto se pode caminhar do elemento A_i para o A_{i+1}, quanto do elemento A_{i+1} para o A_i



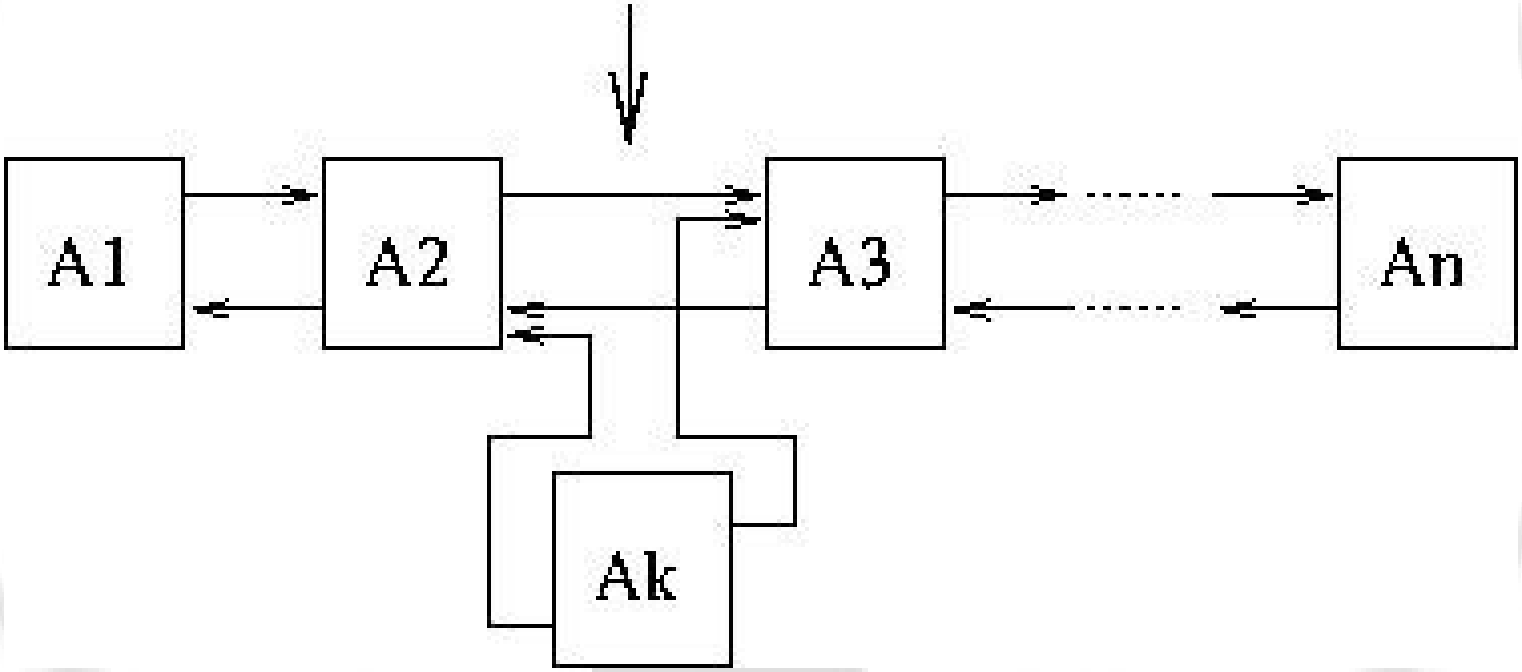
Listas duplamente encadeadas

- Inserção – Localizando a posição



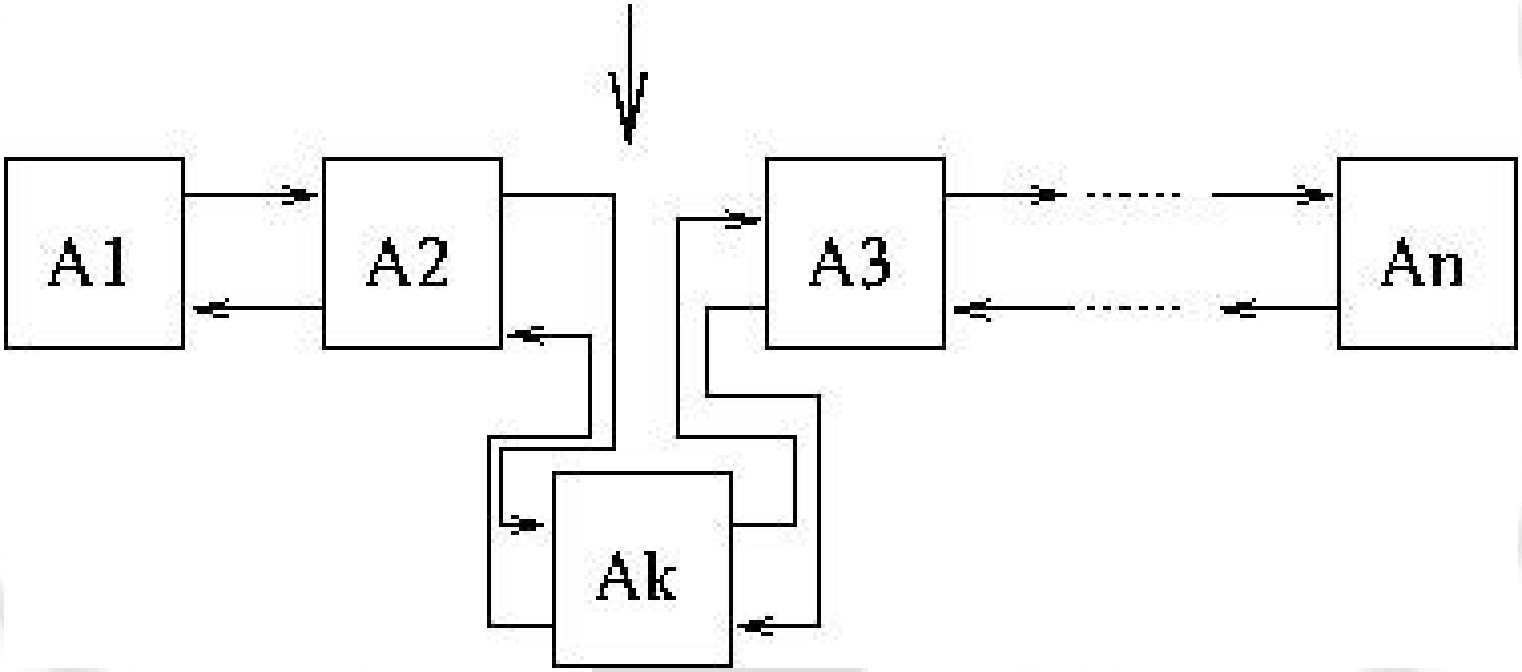
Listas duplamente encadeadas

- Inserção – Reorganizando a lista (1)



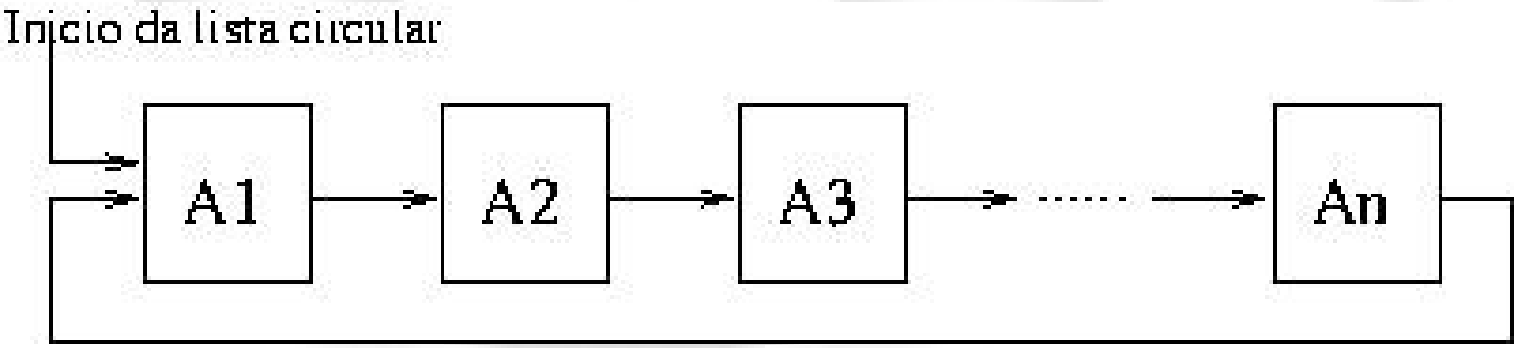
Listas duplamente encadeadas

- Inserção – Reorganizando a lista (2)



Listas circulares

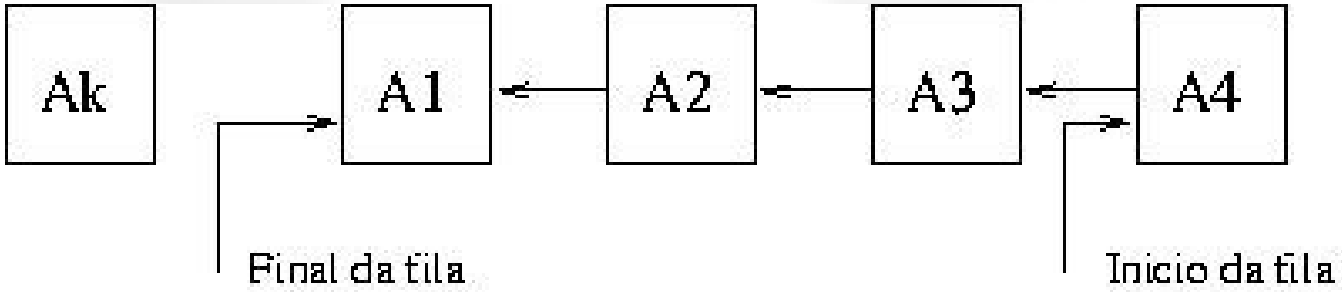
- Outro tipo especial de lista é formado pelas listas circulares
- Nelas existe uma ligação entre entre o último e o primeiro elemento da lista, fechando portanto um ciclo
- Listas circulares são muito usadas na implementação de buffers para entrada de dados



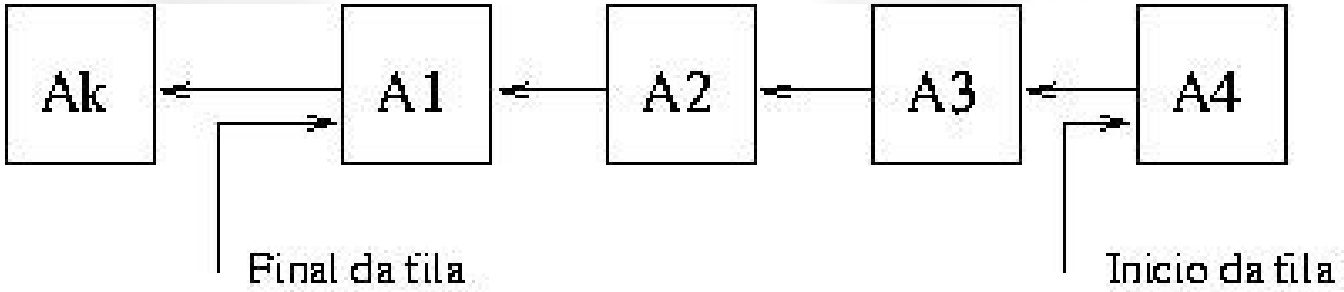
Filas

- São listas em que as operações de remoção e inserção ocorrem sempre em locais específicos
- A inserção é feita sempre no final da lista
- A remoção é feita sempre no início da lista
- Em função disso uma fila assume a condição FIFO

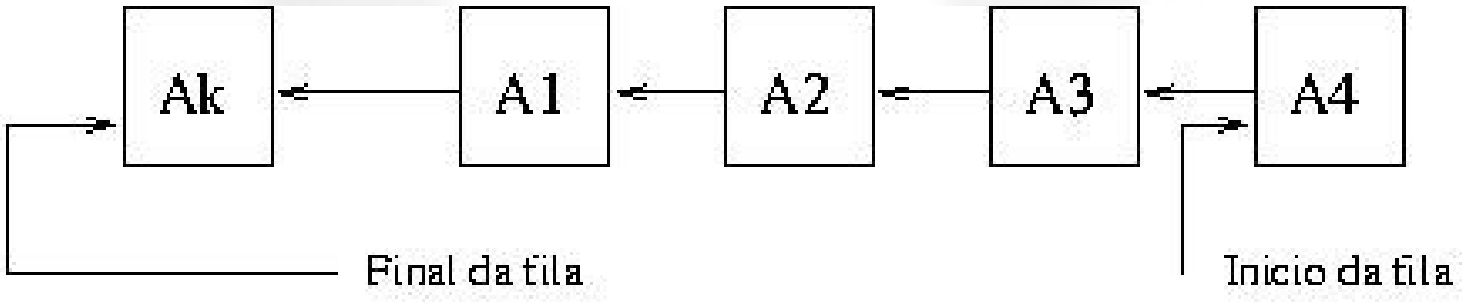
Filas – Inserção (1)



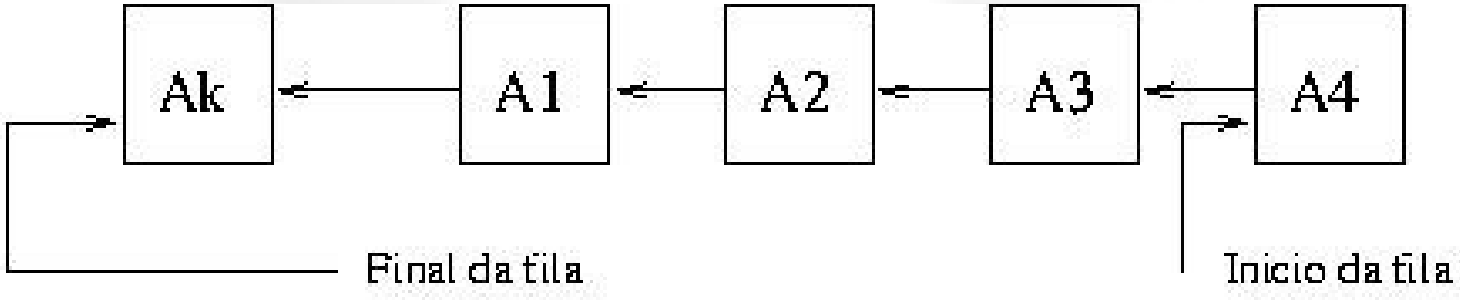
Filas – Inserção (2)



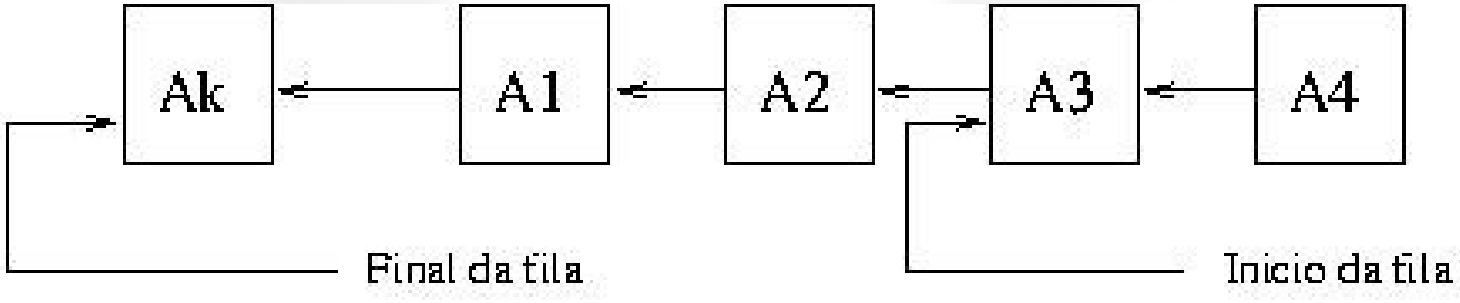
Filas – Inserção (3)



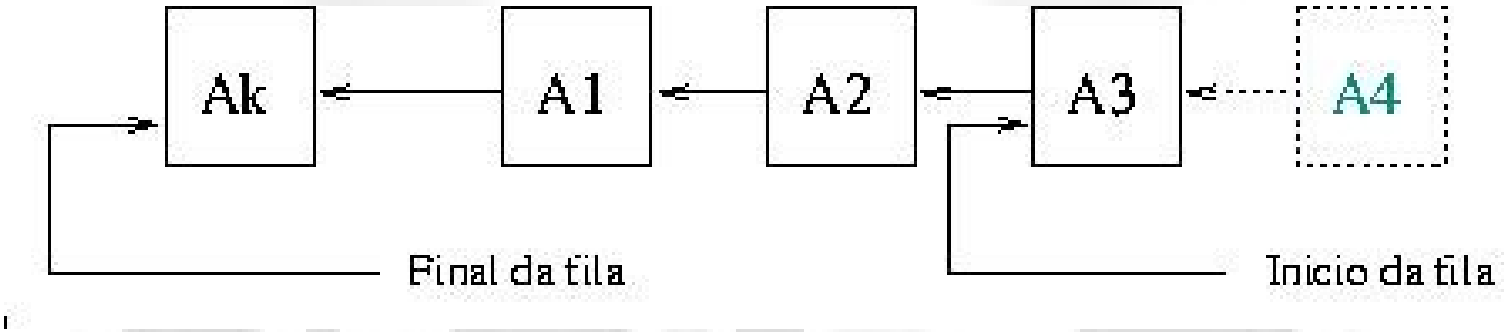
Filas – Remoção (1)



Filas – Remoção (2)



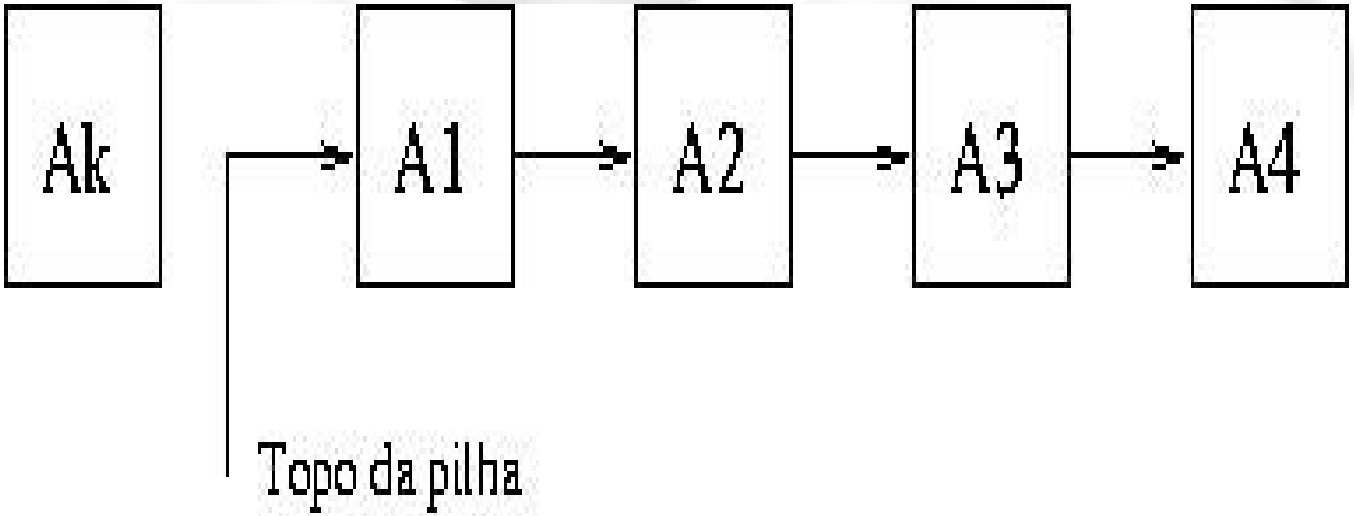
Filas – Remoção (3)



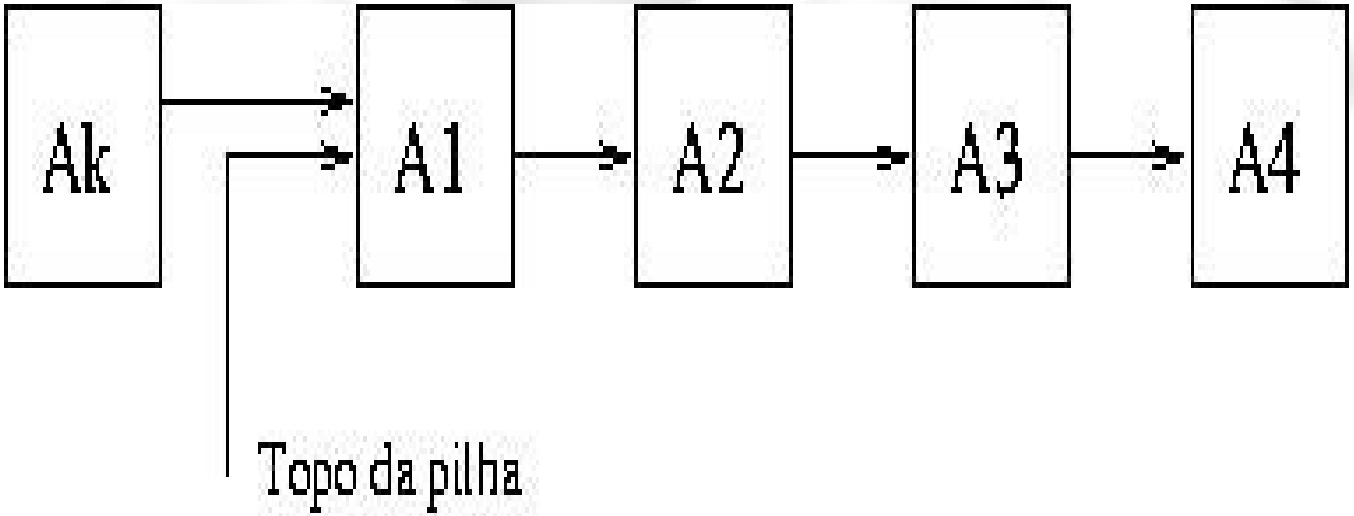
Pilhas

- São listas em que as operações de remoção e inserção ocorrem sempre em locais específicos
- A inserção é feita sempre no início da lista
- A remoção é feita sempre no início da lista
- Em função disso uma fila assume a condição LIFO

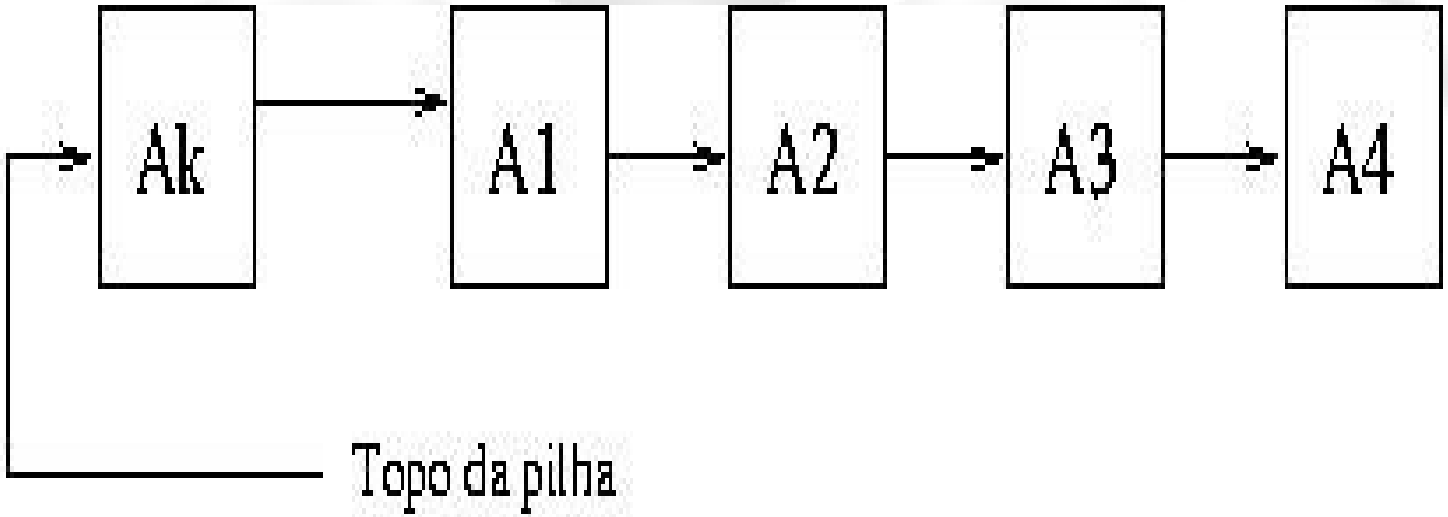
Pilhas – Inserção (1)



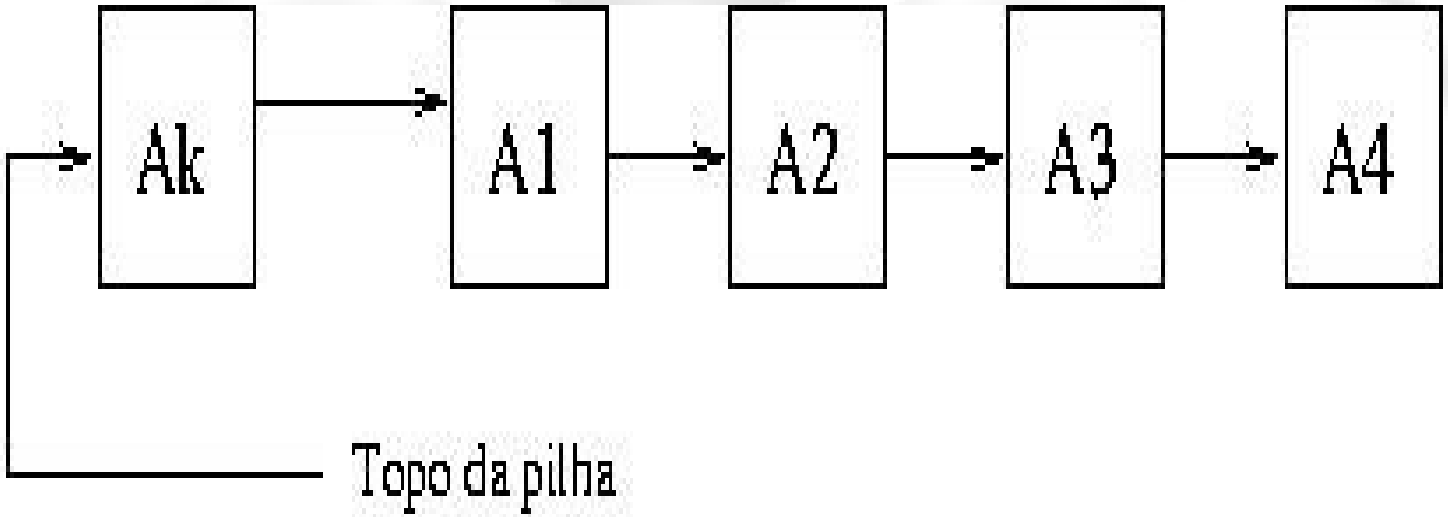
Pilhas – Inserção (2)



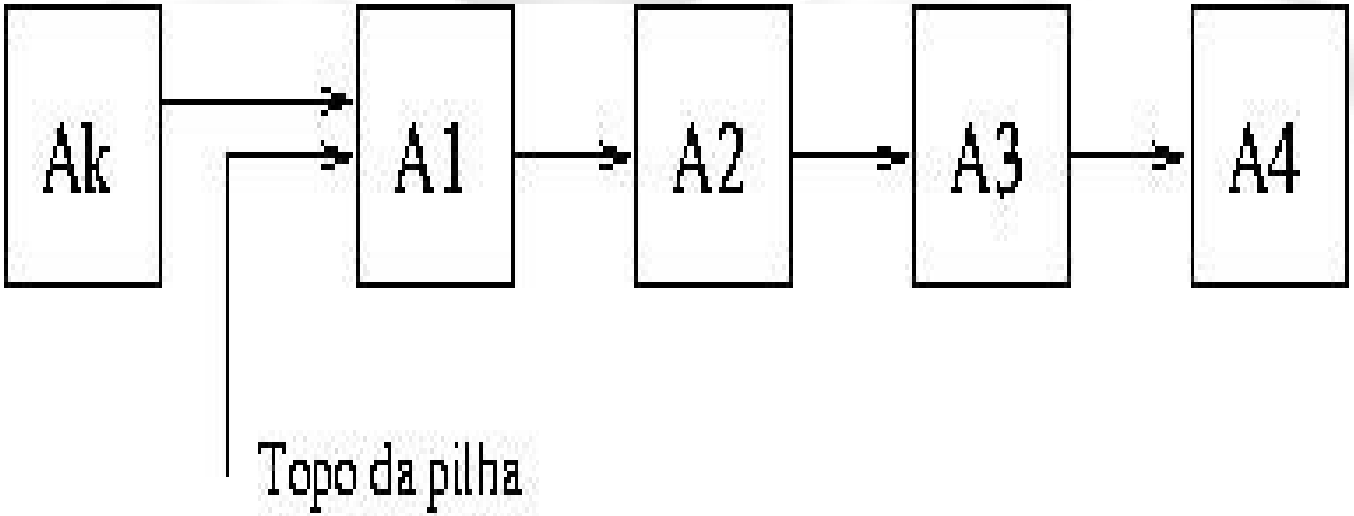
Pilhas – Inserção (3)



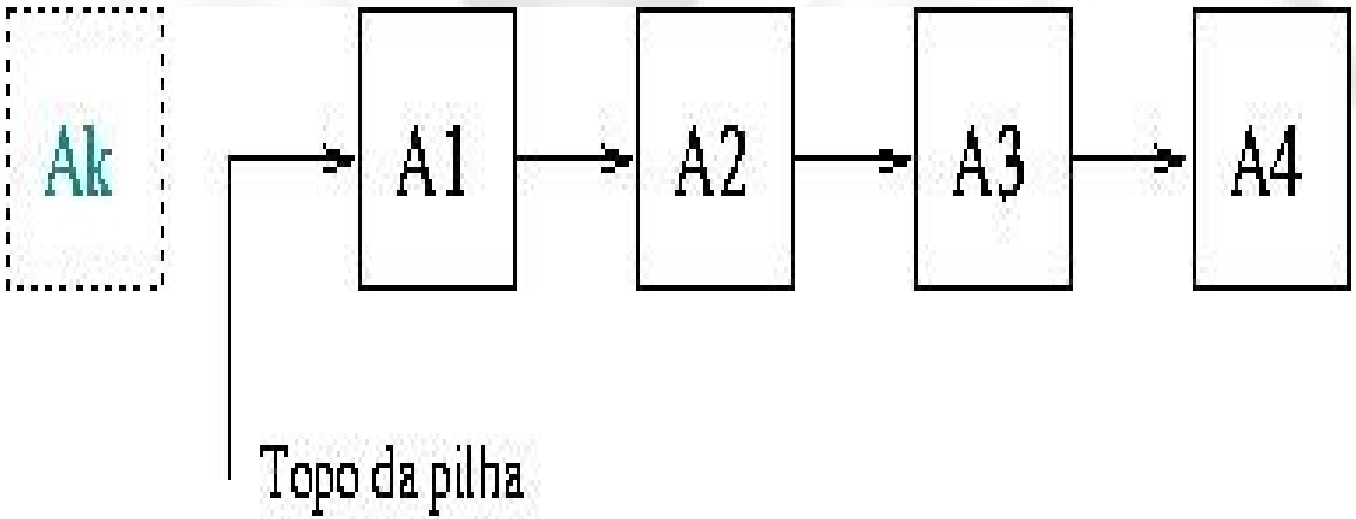
Pilhas – Remoção (1)



Pilhas – Remoção (2)



Pilhas – Remoção (3)



Deques

- Deques (*Double-Ended Queues*) são listas em que as operações de inserção e remoção podem ocorrer apenas em suas extremidades

- Assim, pilhas e filas são especializações de deques

- Os operadores para deques são combinações dos operadores vistos para pilhas e filas

Implementação

- A implementação de qualquer uma das TADs examinadas aqui é relativamente simples

- Existem implementações através de estruturas estáticas (vetores) ou dinâmicas (ponteiros)

- A escolha entre uma e outra forma depende de critérios como espaço a ser consumido e necessidade de velocidade