Estrutura de Dados

Aula 03 – Lista estática

Prof. Dr. Daniel Vecchiato

Agenda

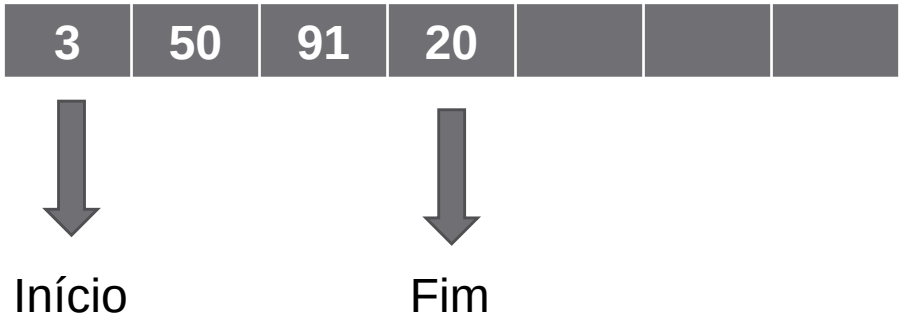
- Introdução
- TAD Lista
- Lista estática
- Exercícios

Introdução

- Em Ciência da Computação, é fundamental o trabalho com conjuntos de dados:
  - Dados de um funcionário;
  - Dados de um cliente;
  - Dados de um produto.
- Esses conjuntos são **dinâmicos**, pois seu tamanho se altera ao longo do tempo.
- Uma ED do tipo lista representa um conjunto de dados organizados em ordem linear.
- Se a lista é representada por um vetor, tem-se o uso de endereços contíguos na memória do computador
  - Esta é a lista estática.
- Se a lista além de conter o dado, contém um ponteiro para o próximo elemento:
  - Esta é a lista dinâmica

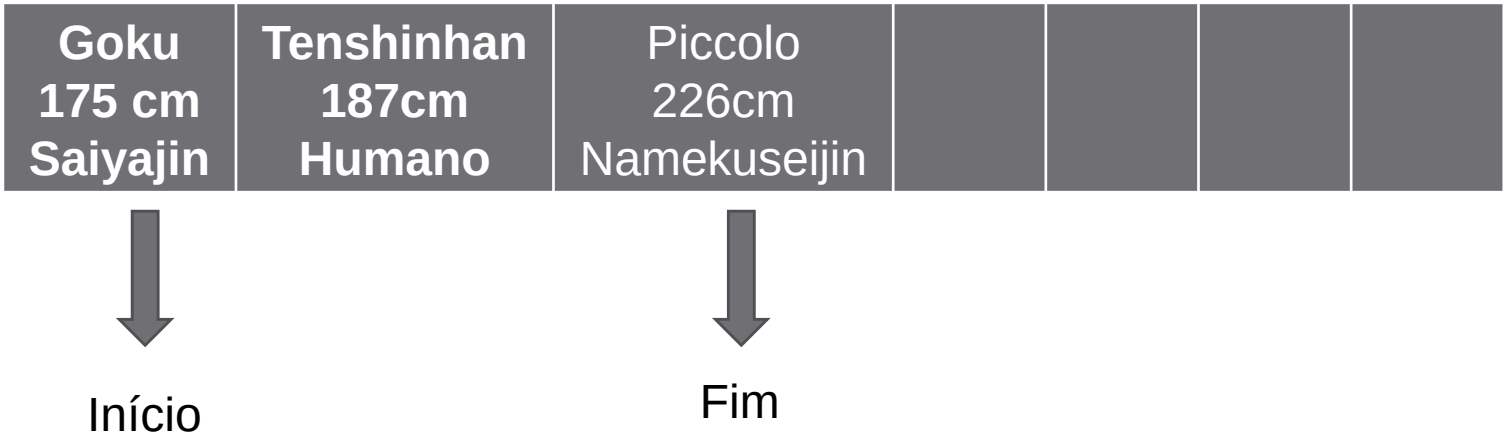
Introdução

- Lista estática homogênea
  - Além do uso do vetor, armazena apenas um dado primitivo



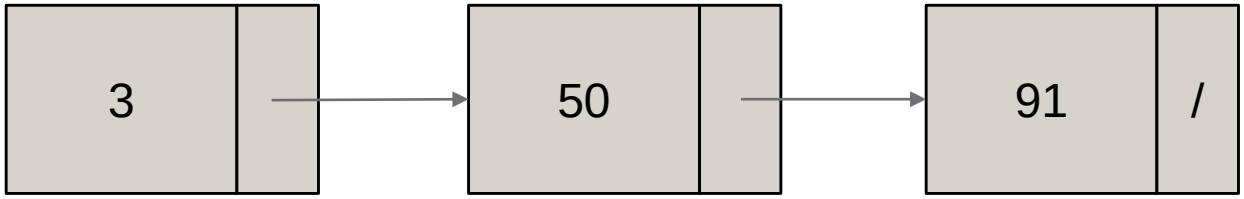
Introdução

- Lista estática heterogênea
  - Além do uso do vetor, armazena um dado composto



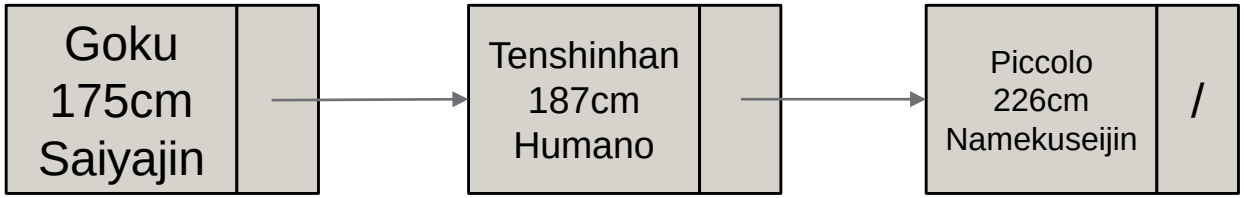
Introdução

- Lista dinâmica homogênea
  - Possui um ponteiro para o próximo elemento e armazena apenas um dado primitivo



Introdução

- Lista dinâmica heterogênea
  - Possui um ponteiro para o próximo elemento e armazena um dado composto



Introdução

- Listas
  - Forma simples de interligar conjuntos
  - Estrutura com operações de:
    - Inserção, remoção e busca, etc.
  - Pode crescer ou diminuir de tamanho durante a execução de um programa
  - Pode ter outras operações como:
    - Concatenação e partição
- Vantagens
  - Manipulação de uma quantidade imprevisível de dados
  - Manipulação de dados de formatos diferentes

Introdução

- Definição
  - Sequência de zero ou mais itens
    - $x_0, x_1, x_2, \ldots, x_{n-1}$, na qual x_i é de um determinado tipo e n representa o tamanho da lista.
    - x_0 é o primeiro item da lista
    - x_{n-1} é o último item da lista
    - x_i precede x_{i+1}
    - x_i sucede x_{i-1}

- Características
  - Sua principal propriedade estrutural envolve as posições relativas dos itens
  - Os itens não necessariamente estão ordenados
  - A lista é linear

Introdução

- TAD
  - Tipo Abstrato de Dado
  - Agrupa a estrutura de dados juntamente com as operações que podem ser feitas sobre esses dados
  - O TAD encapsula a estrutura de dados
    - Os usuários do TAD só tem acesso a algumas operações disponibilizadas sobre esses dados

TAD Lista

- Possíveis operações:
  - Criar uma lista vazia
  - Inserir um novo item
  - Remover um item
  - Localizar um item para consultar ou alterar seu conteúdo
  - Combinar duas ou mais listas em uma única lista
  - Dividir uma lista em duas ou mais listas
  - Fazer uma cópia da lista
  - Ordenar os itens da lista de acordo com um de seus campos
  - Pesquisar a ocorrência de um item com um valor específico

TAD Lista

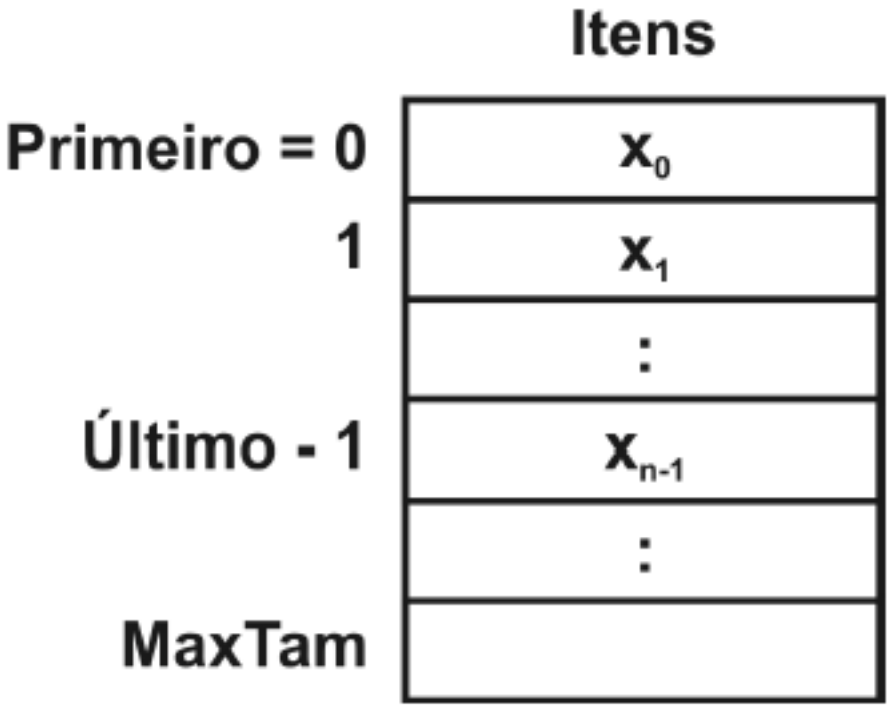
- Protótipo
  - iniciarLista (Lista)
    - cria uma nova lista
  - inserir (Lista, x)
    - insere x no final da lista
  - remover (Lista, p, x)
    - retorna o item x que está na posição p da lista, retirando-o da lista
  - isVazia (Lista)
    - retorna TRUE se a lista estiver vazia, FALSE caso contrário
  - imprimir (Lista)
    - imprime os itens da lista na ordem de ocorrência

TAD Lista

- Implementações
  - Várias estruturas de dados podem ser usadas para representar listas lineares
  - Mais utilizados:
    - Lista estática: Arrays
    - Lista dinâmica: Implementação com ponteiros

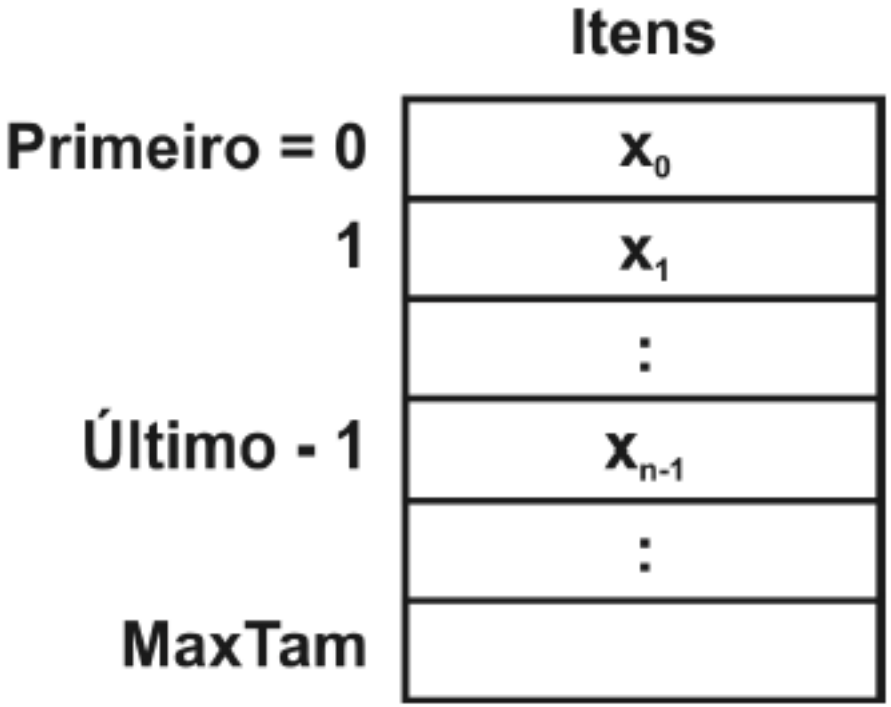
Lista estática

- Visão geral
  - Os itens são armazenados em posições contíguas de memória
  - A lista pode ser percorrida em qualquer direção



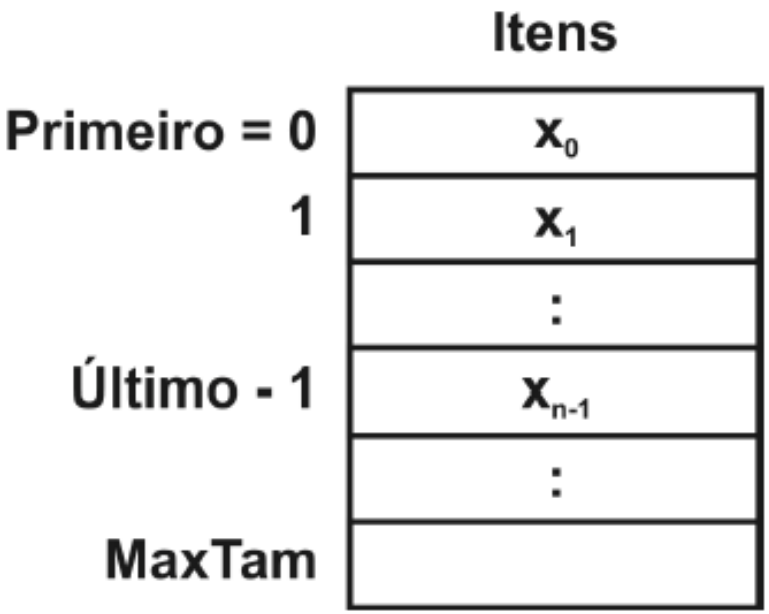
Lista estática

- Visão geral
  - Os itens são armazenados em um array de tamanho suficiente para comportar todos os elementos da lista
  - O campo **Último** aponta para a posição seguinte à do último elemento da lista.



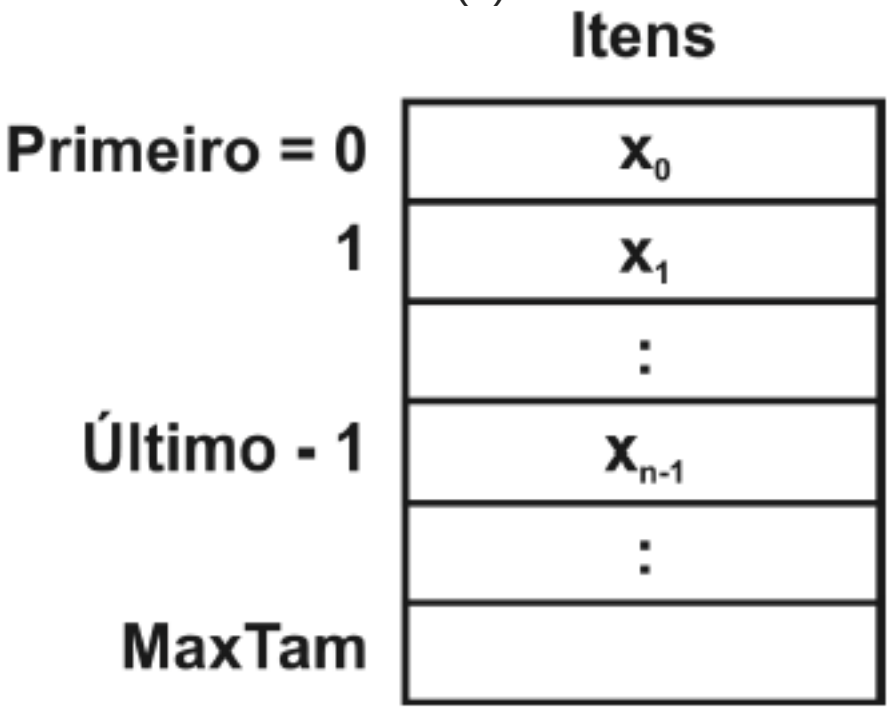
Lista estática

- Visão geral
  - O *i-ésimo* item da lista está armazenado na *(i-1)-ésima* posição do array
    - 0 <= i < Último.
  - A constante **MaxTam** define o tamanho máximo permitido para a lista.



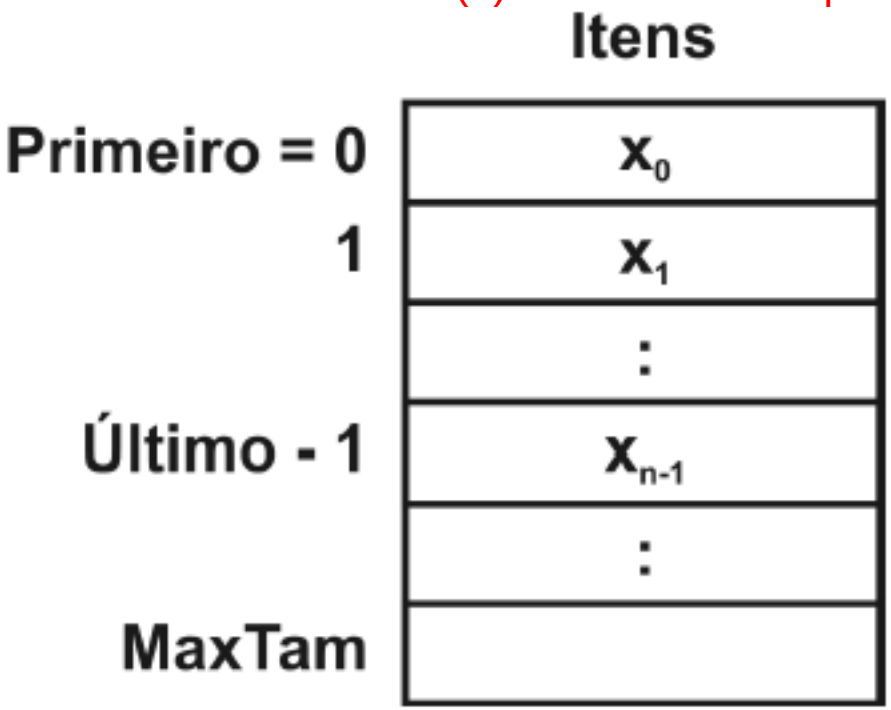
Lista estática

- Visão geral
  - Inserção no final tem custo O(1)
  - Inserção no meio tem um custo O(n)
  - Remoção no meio tem custo O(n)



Lista estática

- Visão geral
  - Inserção no final tem custo O(1)
  - Inserção no meio tem um custo O(n)
  - Remoção no meio tem custo O(n) Por quê?



Lista estática

- Vantagens
  - Acesso a qualquer elemento da lista é feito em tempo O(1)
- Desvantagens
  - Custo para inserir itens da lista pode ser O(n)
  - Custo para remover itens da lista pode ser O(n)
  - Pode ser necessária a realocação do array

Lista estática

```
#include <stdio.h>

#define INICIO 0
#define MAXTAM 500

typedef struct {
    int chave;
} TItem;

typedef struct {
    TItem item[MAXTAM];
    int primeiro, ultimo;
} TLista;
```

Lista estática

```
void iniciarLista (TLista *pLista);
int isVazia (TLista *pLista);
int inserir (TLista *pLista, TItem x);
int remover (TLista *pLista, int p, TItem *pX);
void imprimir (TLista *pLista);
```

Lista estática

```
void iniciarLista (TLista *pLista) {
    pLista->primeiro = INICIO;
    pLista->ultimo = pLista->primeiro;
}

int isVazia (TLista *pLista) {
    return pLista->ultimo == pLista->primeiro;
}

int inserir (TLista *pLista, TItem x) {
    if(pLista->ultimo == MAXTAM )
        return 0; // lista cheia
    pLista->item[pLista->ultimo++] = x;
    return 1;
}
```

Lista estática

```
int remover (TLista *pLista, int p, TItem *pX) {
    if( isVazia (pLista) || p >= pLista->ultimo )
        return 0;

    *pX = pLista->item[p];
    int cont;
    for (cont = p+1; cont <= pLista->ultimo; cont++)
        pLista->item[cont-1] = pLista->item[cont];
    pLista->ultimo--;
    return 1;
}

void imprimir (TLista *pLista) {
    int i;
    printf ("Itens na lista:\n");
    for (i = pLista->primeiro; i < pLista->ultimo; i++)
        printf ("%d\n", pLista->item[i].chave);
}
```

Lista estática

```
int main() {
    TLista lista;
    iniciarLista (&lista);
    printf("Vazia: %s\n", isVazia(&lista) == 1 ? "SIM":"NAO");

    TItem item1, item2;
    item1.chave = 10;
    inserir (&lista, item1);
    item2.chave = -5;
    inserir (&lista, item2);

    imprimir (&lista);
    printf("Vazia: %s\n", isVazia(&lista) == 1 ? "SIM":"NAO");

    TItem itemRemovido;
    remover (&lista, 1, &itemRemovido);
    printf("Item removido: %d\n", itemRemovido.chave);
    imprimir (&lista);
}
```

Exercícios

- Implemente a lista estática baseada em Array
- Acrescente as operações
  - Obter um determinado elemento

```
int get (TLista *pLista, int p, TItem *pX) {
   ...
}
```

    - p: posição desejada
    - pX: estrutura que receberá o elemento obtido
    - Retorna 0 caso não exista a posição, 1 caso contrário
  - Obter quantos elementos há na lista

```
int tamanho (TLista *pLista) {
   ...
}
```

Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura
Daniel Avila Vecchiato

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)