

# Lista Estática

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados I (AE22CP)  
Engenharia de Computação  
Departamento Acadêmico de Informática (Dainf)  
Universidade Tecnológica Federal do Paraná (UTFPR)  
Campus Pato Branco

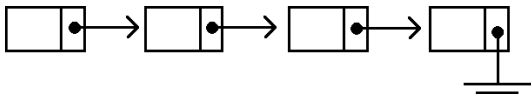
- Listas
- Listas Estáticas
- TAD Listas Estáticas

- Programas operam sobre dados
- Estrutura de dados
- Abstração de informações
  - Tipo abstrato de dados
  - Programação estruturada
  - Programação orientada a objetos
- Conjunto de dados

- Exemplos de abstrações de conjunto de dados alocados sequencialmente (listas)
  - Alocação sequencial contígua

Endereço na memória	3000	3001	3003	3004	
Conteúdo na memória	...				...

- Alocação por encadeamento



## Listas

- Listas são coleções de elementos (variáveis, registros, etc) organizados sequencialmente
  - Agrupam informações referentes a um conjunto de elementos
- As listas estão presentes em várias situações e aplicações do nosso cotidiano
  - Compras
  - Lista de presença
  - Contatos no smartfone
  - Livros

- Aplicações
  - Adequadas para aplicações em que não é possível prever a demanda por memória
    - Gerenciamento de memória
    - Simulações
    - Manipulação simbólica
    - Processamento de imagens
    - Entre outras
- Listas podem ser do tipo linear ou não-linear (generalizada)
- Tipos especiais de listas
  - Fila
  - Pilha

- Uma lista linear é uma sequência de 0 ou mais itens  $x_1, x_2, \dots, x_n$ 
  - $n$  é o tamanho da lista
  - $x_i$  é o  $i$ -ésimo elemento da lista
  - $x_1$  é o primeiro item da lista e  $x_n$  o último
  - $x_i$  sucede  $x_{i-1}$  e precede  $x_{i+1}$
  - Se  $n = 0$ , então a lista é vazia



- Operações básicas em uma lista:
  - Criar uma lista vazia
  - Inserir um elemento
  - Remover um elemento
  - Verificar se a lista está vazia
  - Procurar um elemento
  - Concatenar duas ou mais listas

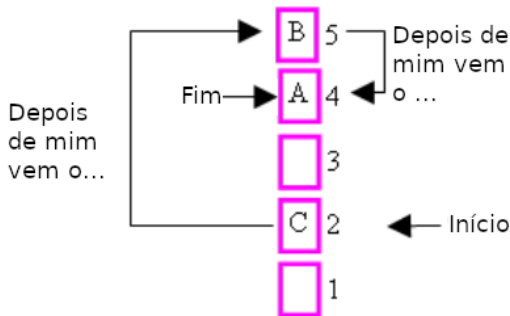
- Operações básicas em uma lista:
  - Dividir uma lista em duas ou mais listas
  - Ordenar a lista
  - Imprimir todos os elementos
  - Retornar referência (e.g. primeiro ou próximo item da lista)
  - Liberar lista
  - Etc

- Alocação sequencial (contígua): elementos são alocados em sequência (sequência "física")
  - Geralmente são utilizadas estruturas estáticas (vetores)

0	1	2	3	4	5	6	7
A	B	C	D				

- Na lista acima:
  - 'A' é o primeiro elemento
  - 'D' é o último elemento
  - Por mais que a lista tenha a capacidade de alocar até 8 elementos, apenas 4 posições foram utilizadas em sequência
  - Mais elementos podem ser colocados na estrutura acima

- Alocação encadeada: elementos não estão necessariamente em posições adjacentes de memória (sequência "lógica" ou "virtual")
  - Geralmente são utilizadas estruturas dinâmicas (apontadores)

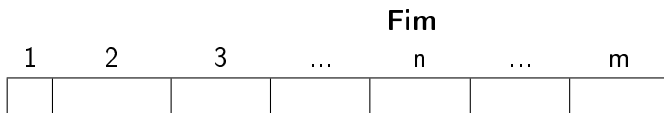


## Listas Estáticas

- Os itens dessa estrutura são armazenados em posições contíguas da memória
- A lista estática pode ser percorrida, linearmente, em qualquer direção
- Armazenamento em arranjos (*arrays*/vetores)

Endereço na memória	3000	3001	3003	3004	
Conteúdo na memória	...				...

- Uma forma de implementação de listas estáticas, além de considerar um vetor, também podem ser definidas
  - Uma variável para indicar a posição do último elemento

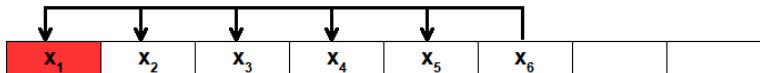


# Listas Estáticas

- Um novo item pode ser inserido no final da lista com custo constante



- No entanto, a remoção de item requer o deslocamento de itens para preencher o espaço vazio





- Vantagens
  - Simples implementação
  - Economia de memória
- Desvantagens
  - Custo para retirar itens
  - Caso a lista atinja o limite de armazenamento, não é possível realocar memória

## TAD Listas Estáticas

- Geralmente, em uma lista é necessária as seguintes operações
  - Criar uma lista vazia
  - Inserir um item
  - Remover um item
  - Acessar um item
  - Verificar se a lista está vazia
  - Verificar se a lista está cheia
  - Imprimir a lista
  - Liberar lista

- Primeiro passo: definir arquivo .h

```
// List.h
#define MAX_SIZE 100 // tamanho máximo da lista

typedef struct Lista Lista;
```

- Primeiro passo: definir arquivo .h

```
// lista.h
Lista* criar_lista();

int vazia(Lista *l);

int cheia(Lista *l);

int buscar(Lista *l, int chave);

int inserir(Lista *l, int chave);

int remover(Lista *l, int chave);

void imprimir(Lista *l);

void liberar(Lista *l);
```

- Segundo passo: definir arquivo .c

```
// lista.c
#include "lista.h"

struct Lista{
    int item[MAX_SIZE];
    int qtd; // quantidade de elementos que foram
    colocados na lista
};

List* criar_lista(){
    Lista *l = (Lista*) malloc(sizeof(Lista));
    l->qtd = -0;

    return l;
}
```

```
int vazio(Lista *l){
    return (l == NULL) || (l->qtd == 0);
}

int cheio(Lista *l){
    return (l->qtd == MAX_SIZE);
}

int buscar(Lista l, int chave){
    int i;

    for (i = 0; i < l->qtd; i++)
        if (key == l->item[i])
            return i;

    return -1;
}
```

# TAD Listas Estáticas

```
int inserir(Lista *l, int chave){  
    if (!cheio(l)){  
        l->item[l->qtd] = chave;  
        l->qtd++;  
        return 1;  
    }  
  
    return 0;  
}
```



```
int remover(Lista *l, int chave){
    int i;
    int p = buscar(*l, chave);

    if (p >= 0){
        for (i = p; i < l->qtd - 1; i++)
            l->item[i] = l->item[i + 1];

        l->qtd--;
        return 1;
    }

    return 0;
}
```





```
void imprimir(List *l){
    int i;

    for (i = 0; i < l->qtd; i++)
        printf("%d\n", l->item[i]);
}

void liberar(List *l){
    free(l);
}
```

- Os arquivos com implementação mais detalhada do TAD de listas estáticas estão disponíveis no Github: [https://github.com/jefferson-oliva/material\\_grad/tree/main/AE22CP-Algoritmos-1/aula\\_06\\_lista\\_est%C3%A1tica](https://github.com/jefferson-oliva/material_grad/tree/main/AE22CP-Algoritmos-1/aula_06_lista_est%C3%A1tica)

- Exercício: aproveitando o TAD anterior, faça:
  - Implemente uma função que concatena duas listas
  - Altere o TAD de forma que os itens devam estar ordenados
  - Implemente uma função que intercale duas listas em uma terceira de forma ordenada

-  Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Clifford, S.  
*Algoritmos: teoria e prática.*  
Elsevier, 2012.
-  Pereira, S. L.  
*Estrutura de Dados e em C: uma abordagem didática.*  
Saraiva, 2016.
-  Szwarcfiter, J.; Markenzon, L.  
*Estruturas de Dados e Seus Algoritmos.*  
LTC, 2010.
-  Tenenbaum, A.; Langsam, Y.  
*Estruturas de Dados usando C.*  
Pearson, 1995.



Ziviani, M.

*Projetos de Algoritmos: com implementações em Pascal e C.*  
Thomson, 2004.