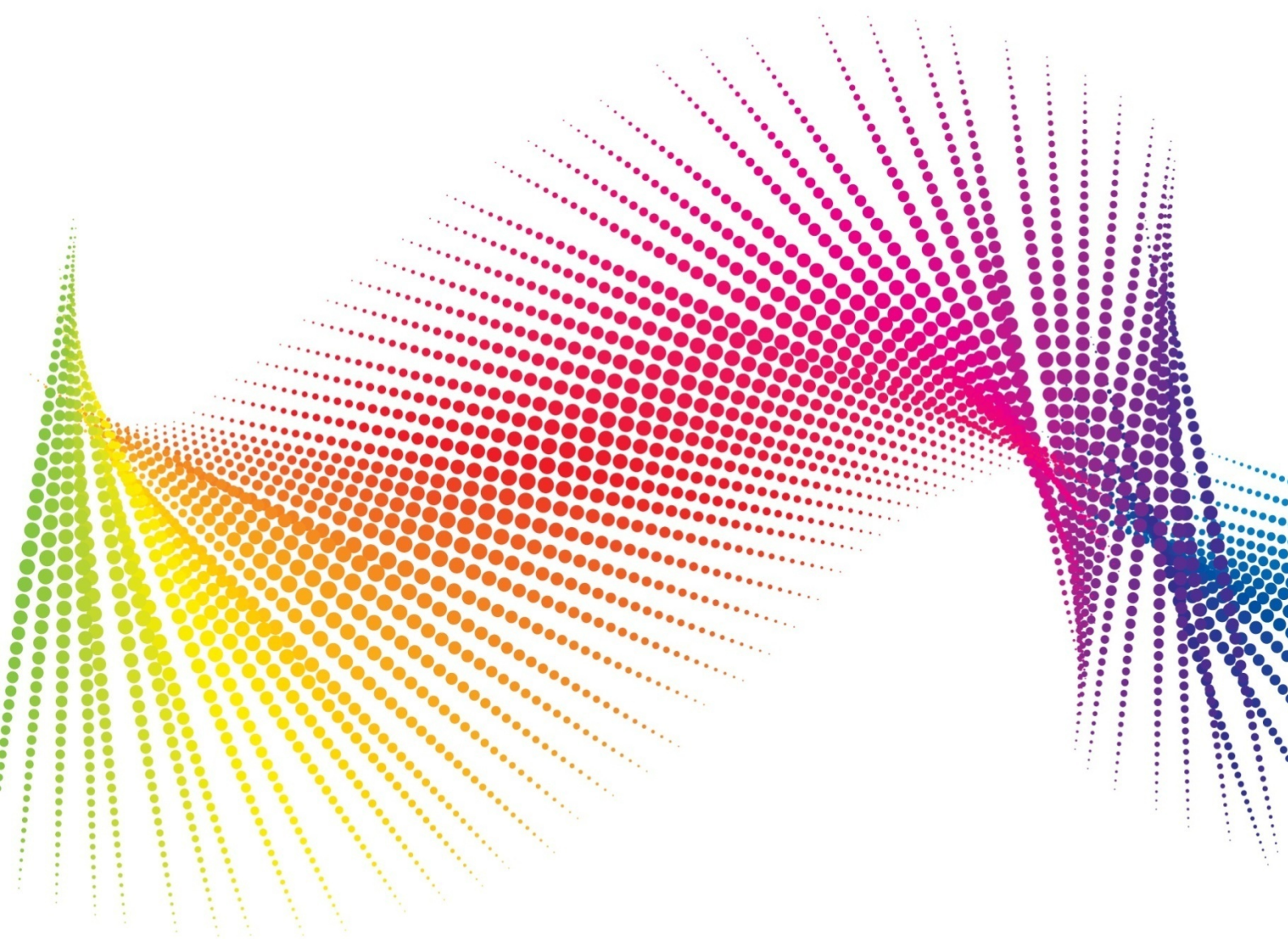


# Estrutura de Dados

Aula 19



Este material é parte integrante da disciplina oferecida pela UNINOVE.

O acesso às atividades, conteúdos multimídia e interativo, encontros virtuais, fóruns de discussão e a comunicação com o professor devem ser feitos diretamente no ambiente virtual de aprendizagem UNINOVE.

Uso consciente do papel.

Cause boa impressão, imprima menos.

## **Aula 19: Listas encadeadas com alocação dinâmica**

**Objetivo:** Estudar uma estrutura de dados muito importante: as listas encadeadas com alocação dinâmica.

### **Conceitos básicos sobre tipo abstrato de dados e estruturas de dados**

Caros alunos, nesta aula vamos introduzir o conceito de alocação dinâmica e estudar as listas encadeadas, que é um tipo abstrato de dados. Acesse o AVA e assista à animação para revisarmos alguns conceitos antes de iniciarmos o tema. Esta animação faz parte da sequência desta aula e, portanto, é essencial para a sua aprendizagem.

Nesta aula, iremos estudar as listas encadeadas com alocação dinâmica, que, acompanhadas das funções de inserção, retirada e localização de nós, formam um tipo abstrato de dados. A animação ajudou-o a entender o que é um tipo abstrato de dados, certo? Pois bem, esta é mais uma importante estrutura de dados em memória principal que nos ajudará a resolver algumas classes de problemas.

### **Alocação sequencial x alocação dinâmica**

Uma lista linear é um tipo abstrato de dados capaz de armazenar um conjunto de dados os quais, geralmente, os dados obedecem a uma sequência.

A implementação de uma lista linear pode ser feita de forma estática, usando-se vetores, ou dinâmica, usando-se ponteiros. Cada tipo apresenta vantagens e desvantagens.

Para usarmos uma lista linear estática é necessário conhecermos o número máximo de elementos a ser armazenado; o armazenamento destes ocupa posições sequenciais na memória.

Na implementação dinâmica são alocados espaços na memória na medida da necessidade e as posições de memória não obedecem a uma sequência. Além disso, o programador é que deve fazer a “ligação” entre os elementos para determinar sua sequência.

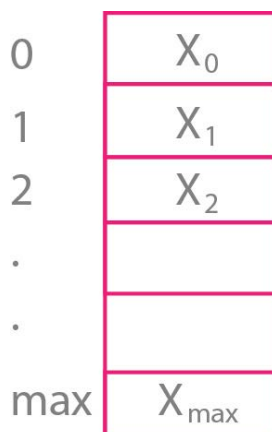
## Listas lineares: alocação estática

### Vantagens:

- Economia de memória e grande facilidade para percorrer entre os elementos e localizá-los, pois o acesso pode ser direto.
- Elementos são armazenados sequencialmente na estrutura e na memória.

### Desvantagens:

- Custo alto para remover itens, pois a retirada de um elemento geralmente implica o reposicionamento de diversos outros elementos do vetor.
- Custo alto para inserir elementos de forma ordenada, pois o deslocamento de um elemento geralmente implica o deslocamento de diversos outros elementos do vetor.
- Em aplicações que não se sabe o tamanho máximo da lista, a implementação estática é problemática, pois o vetor é declarado inicialmente no programa com tamanho pré-definido; caso se atinja a capacidade máxima do vetor, o programa não pode aceitar mais elementos na lista.



## Listas lineares: alocação dinâmica

### Vantagens:

- Não é necessário se saber inicialmente o tamanho da lista, pois cada vez que um novo elemento entra na lista, um novo espaço na memória é alocado para seu armazenamento.

- Utiliza-se apenas o espaço de memória dos elementos alocados.
- Quando um elemento é retirado da lista, o espaço que ele ocupava pode ser deslocado, ou seja, liberado para que possa ser reutilizado.

**Desvantagens:**

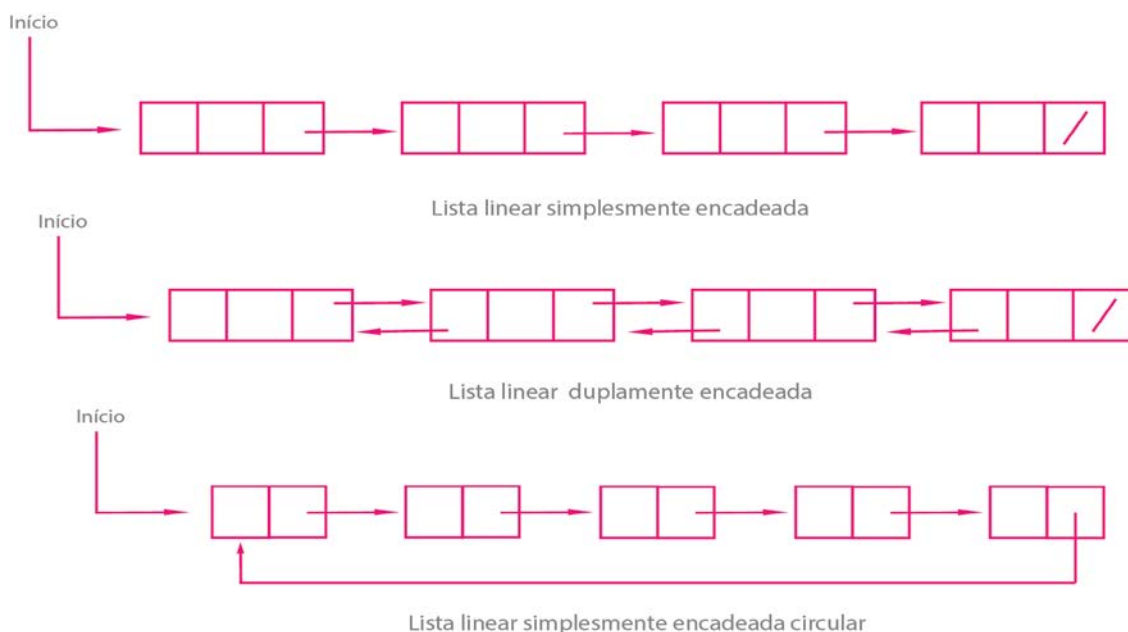
- Maior complexidade para controle do encadeamento dos elementos, pois o programador é o responsável por fazer a ligação ou encadeamento dos elementos.

**Listas encadeadas com alocação dinâmica**

Podemos criar uma lista linear com os elementos ligados por ponteiros. Este tipo de lista chama-se lista encadeada. Uma lista encadeada é uma forma especial de se agrupar dados, em que cada item possui uma referência para o próximo item, como se fosse uma corrente, com cada item sendo um dos elos. Costuma-se chamar esses “elos” ou itens de nós ou nodos.

As listas lineares encadeadas apresentam uma subclassificação em relação ao número e funcionamento dos ponteiros que são utilizados na estrutura:

- LLSE: Lista Linear Simplesmente Encadeada
- LLDE: Lista Linear Duplamente Encadeada
- LLC: Lista Linear Circular



### Lista encadeada com alocação dinâmica – muito simples

Como vimos, caro aluno, quando representamos uma lista linear por meio de um vetor, a relação de precedência é indicada pelo fato de  $\text{vet}[x]$  e  $\text{vet}[x+1]$  ocuparem posições sequenciais no vetor e, conseqüentemente, posições sequenciais na memória. Este tipo de implementação estática utiliza **alocação sequencial** de memória.

Mas, vimos também, que podemos usar uma implementação com alocação dinâmica. Para isso, devemos representar a relação de precedência entre os elementos. Isso pode ser feito de forma muito simples: basta criarmos uma estrutura (struct) que seja capaz de armazenar o elemento e tenha um campo que contenha o endereço de memória do próximo elemento de lista, ou seja, **um apontador ou ponteiro**. Dentro desta nova concepção, podemos criar a implementação com **alocação dinâmica** de memória.

Para sabermos como criar uma estrutura que realize este tipo de implementação com alocação dinâmica, leia o material complementar. Este material faz parte da sequência da aula, essencial para seu aprendizado.

## Programa que implementa uma lista encadeada muito simples

```
//llesimples.c: mostra a implementação de uma llse
// de forma rudimentar ou muito simples
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

//declara o nó e apenas 2 ponteiros: início e aux
typedef struct nodo {    int info;
                        struct nodo *prox;
                        } no;

no *inicio, *aux;

int main()
{    system("cls");
    system("color 1e");
    printf ("\nImplementa uma lista encadeada muito simples\n");
    printf ("\n_____ \n");

    // inicialmente a lista está vazia, então o ponteiro início aponta para "vazio"
    inicio = NULL;

    // aloca espaço para o primeiro elemento da lista encadeada
    aux = malloc(sizeof(no));
    printf("\nDigite um numero para entrar na lista encadeada: ");
    scanf("%i",&aux->info);
    aux->prox=NULL;

    //faz início apontar para o primeiro no da lista
    inicio = aux;

    // aloca espaço para o segundo nó da lista encadeada
    aux = malloc(sizeof(no));
    printf("\nDigite outro numero p/ entrar na lista encadeada:");
    scanf("%i",&aux->info);
    aux->prox =NULL;

    //faz o encadeamento do nó na lista: início aponta para o primeiro nó
    // e este deve apontar para o segundo nó
    inicio->prox= aux;

    // aloca espaço para o terceiro nó da lista encadeada
    aux = malloc(sizeof(no));
    printf("\nDigite outro numero p/ entrar na lista encadeada: ");
```

```
scanf("%i",&aux->info);
aux->prox =NULL;
```

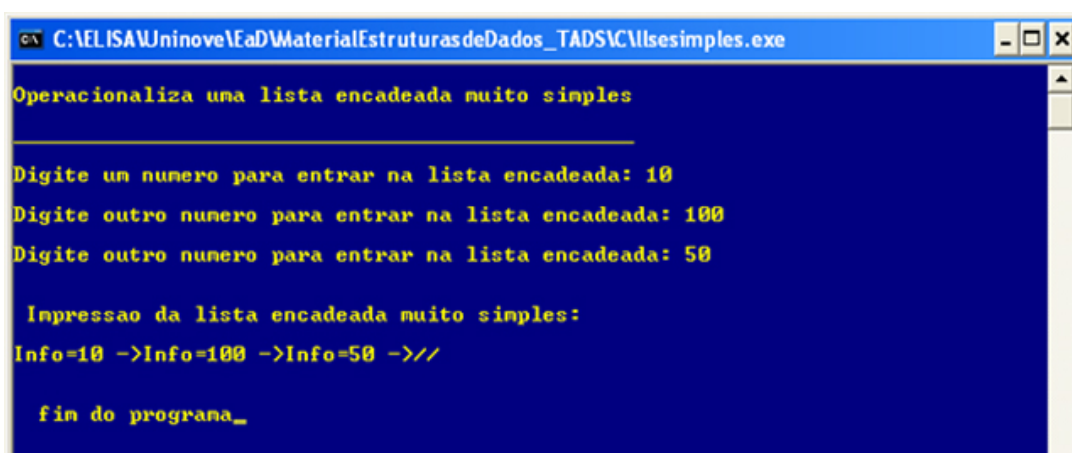
**//faz o encadeamento nó na lista: início aponta para o primeiro nó**  
**// que aponta para o segundo nó que deve apontar para o terceiro nó**  
 inicio->prox->prox = aux;

**//foi criada uma lista encadeada simples com apenas 3 nós**  
**//agora esta lista será impressa na tela**

```
printf("\n\nImpressao da lista encadeada muito simples: \n\n");
aux = inicio;
while (aux != NULL)
{
    printf("Info=%i ->", aux->info);
    aux=aux->prox;
}
printf("//\n\n");

printf ("\n fim do programa");
getch();
return 0;
}
```

Caro aluno, certamente agora que você entendeu bem este programa que mostra como criar uma lista encadeada com alocação dinâmica muito simples, você vai implementar e testar este programa. Confira a saída do programa:



```
C:\ELISA\Uninove\FaD\MaterialEstruturasdeDados_TADS\VC\llesimples.exe

Operacionaliza uma lista encadeada muito simples

Digite um numero para entrar na lista encadeada: 10
Digite outro numero para entrar na lista encadeada: 100
Digite outro numero para entrar na lista encadeada: 50

Impressao da lista encadeada muito simples:
Info=10 ->Info=100 ->Info=50 ->/

fim do programa_
```

O problema evidente com esta implementação de lista encadeada com alocação dinâmica apresentada é que os nós ficam dependentes do primeiro nó apontado por `inicio`, e para aumentar ou percorrer a lista os encadeamentos vão ficando muito longos.



`inicio->prox->prox->prox = aux;`

Então, é necessária uma sistematização das rotinas para INSERÇÃO, REMOÇÃO, LOCALIZAÇÃO e IMPRESSÃO dos nós de uma lista encadeada com alocação dinâmica.

Na próxima aula estas rotinas serão apresentadas, já organizadas dentro do programa.

Agora, caro aluno, vamos praticar resolvendo os exercícios propostos. **Leia a lista, resolva os exercícios e verifique seu conhecimento.** Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

## REFERÊNCIAS

SCHILDT, H. *C completo e total*. São Paulo: Makron Books, 1997.

TENEMBAUM, Aaron M., et al. *Estruturas de dados usando C*. São Paulo: Pearson Makron Books, 1995.