LISTAS

Profa. Fabrícia Damando Santos
fabriciadamando@gmail.com

Listas

- Uma lista linear é um agregado linear de elementos, ou seja, é uma estrutura que permite representar um conjunto de dados de forma a preservar a relação de ordem linear entre eles

- Uma lista é um conjunto de nós.

- Quando trabalhamos com listas, cada nó deve conter o índice de seu sucessor.

LISTAS

- São estruturas formadas por um conjunto de dados de forma a preservar a relação de ordem linear entre eles.

- Uma lista é composta por nós, os quais podem conter, cada um deles, um dado primitivo ou composto.

Listas

- Estruturas de dados de manipulação simples
- Agrupa elementos que se relacionam entre si
  - Exemplo: informações de funcionários
- Conjunto de nós
- Operações comuns: *busca, inclusão e remoção*

LISTAS

- **Sendo:**
  - L1 - 1º elemento da lista
  - L2 - Sucessor de L1
  - Ln-1 - Antecessor de Ln-1
  - Ln - Último elemento da lista.

Exemplo de listas

- **Exemplos de Lista:**
  - Lista Telefônica
  - Lista de clientes de uma agência bancária
  - Lista de setores de disco a serem acessados por um sistema operacional
  - Lista de pacotes a serem transmitidos em um nó de uma rede de computação de pacotes

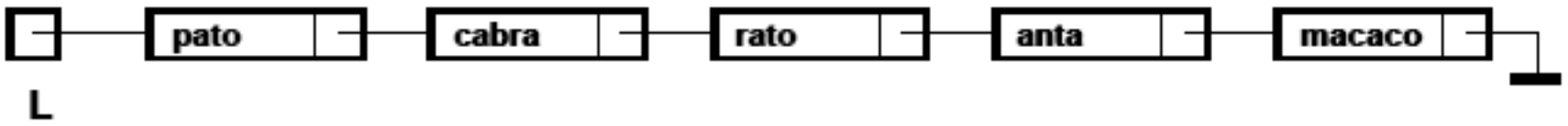
Operações realizadas com listas

- **Operações Realizadas com Listas:**
  - Criar uma lista vazia
  - Verificar se uma lista está vazia
  - Obter o tamanho da uma lista
  - Obter/modificar o valor do elemento de uma determinada posição na lista
  - Obter a posição de elemento cujo valor é dado
  - Inserir um novo elemento após (ou antes) de uma determinada posição na lista
  - Remover um elemento de uma determinada posição na lista
  - Exibir os elementos de uma lista
  - Concatenar duas listas

Representação

- Sequencial
  - São armazenados endereços sequenciais podendo ser armazenados por um vetor

- Encadeada
  - Sequencia de elementos encadeados por um ponteiro, onde cada elemento ***deve conter*** além do ***dado,*** uma ***referência para o próximo*** elemento da lista



LISTA ENDADEADA

- Uma lista encadeada é uma seqüência de ***células***; cada uma contém um objeto de algum tipo e o endereço do célula seguinte.

- Vamos supor que nesta página que os objetos armazenados nas células são do tipo int.

- A estrutura de cada célula de uma tal lista pode ser definida assim:

```
struct registro
 {
   int dado;
   struct registro *prox;   };
```

LISTAS ENCADEADAS

- ***Observe, na figura ao lado:***
- O vetor de 50 posições que possui os campos **info** e **prox**.
- O primeiro campo armazena a informação dos elementos da lista (nesse exemplo, números inteiros) e o segundo armazena o índice do próximo nó da lista.
- O último nó da lista irá conter -1 no campo prox, indicando que é o final da lista.

		info	prox
	0	26	-1
	1	11	9
	2	5	15
lista 4 =	3	1	24
lista 2 =	4	17	0
	5	13	1
	6		
	7	19	18
	8	14	12
	9	4	21
	10		
lista 3 =	11	31	7
	12	6	2
	13		
	14		
	15	37	23
lista 1 =	16	3	20
	17		
	18	32	-1
	19		
	20	7	8
	21	15	-1
	22		
	23	12	-1
	24	18	5
		...	...
	49		

- No exemplo anterior, as variáveis inteiras lista 1, lista 2, lista 3 e lista 4 são inteiros que representam o início de quatro listas, respectivamente

- Lista 1: 3, 7, 14, 6, 5, 37, 12
- Lista 2: 17, 26
- Lista 3: 31, 19, 32
- Lista 4: 1, 18, 13, 11, 4, 15

- Inicialmente, todos os nós estão **sem uso** porque nenhuma lista foi formada.
- Dessa maneira, eles devem ser colocados em uma lista de nós livres.
- Assim, é preciso criar uma lista de nós livres, onde cada nó livre aponta para o seu sucessor no vetor.
- Portanto, o primeiro nó disponível estará na posição 0, o segundo na posição 1 e assim por diante.
- O último nó da lista, que estará na última posição do vetor, terá valor -1 no campo **prox**.

- Do mesmo modo que existiam quatro variáveis para controlar as listas 1, 2, 3 e 4, anteriormente, também deverá haver uma outra variável para controlar a lista de nós livres.

- Quando for necessário, por exemplo, inserir um elemento em uma determinada lista, a lista de nós livres deverá ser consultada e a posição disponível ser obtida de lá.

- Quando o nó não for mais necessário, ele deverá retornar para essa lista.

LISTAS DINÂMICAS

- Ao invés de usar uma implementação estática, podemos solucionar os problemas da implementação em vetor usando alocação dinâmica
  - quando um nó for necessário, o armazenamento ficará reservado para ele e, quando não for mais necessário, o armazenamento será liberado.
  - Dessa forma, o armazenamento para nós não mais em uso ficará disponível para outro propósito.

LISTAS DINÂMICAS

- Além disso, não é estabelecido um limite predefinido sobre o número de nós.

- Enquanto houver armazenamento suficiente disponível para o programa como um todo, parte desse armazenamento poderá ser reservada para uso como um nó.

LISTAS SIMPLESMENTE ENCADEADA

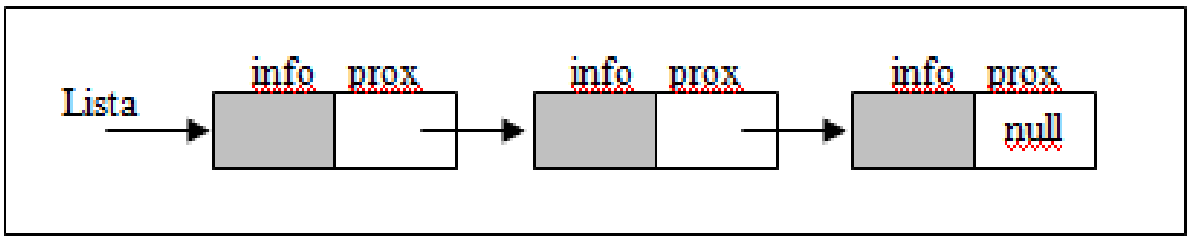
- Uma lista simplesmente encadeada é uma seqüência de objetos alocados dinamicamente, cada qual fazendo referência ao seu sucessor na lista.

- Para isso, cada item da lista, ou seja, cada objeto, deverá conter pelo menos dois campos: um de **informação** e outro de **endereço seguinte**.

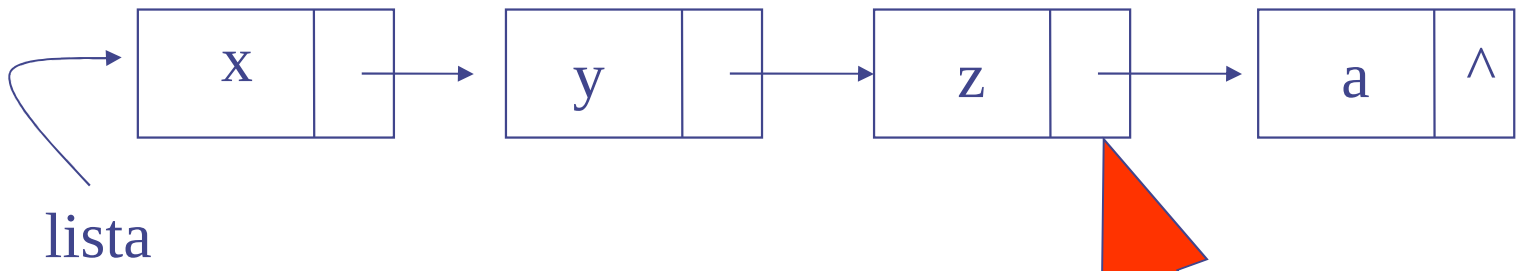
- O campo de informação armazena o real elemento da lista.

- O campo endereço seguinte contém o endereço do próximo objeto na lista.

- A lista ligada inteira é acessada a partir de uma variável **Lista**, por exemplo, que aponta para o primeiro nó da lista.



LISTA SIMPLESMENTE ENCADEADA

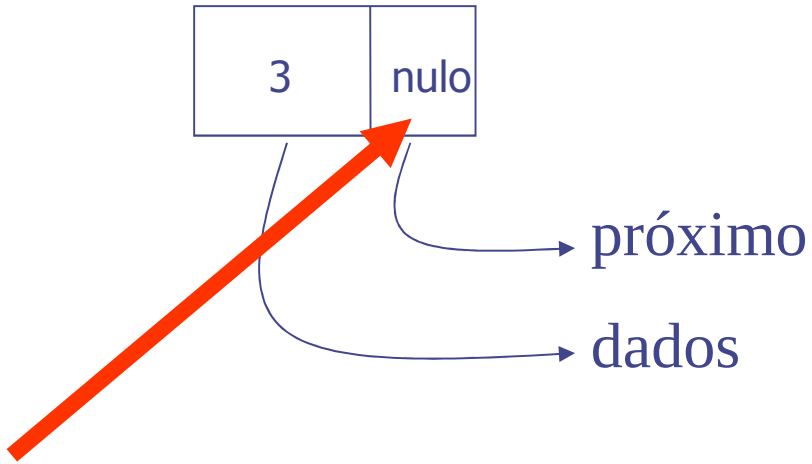


- os componentes da lista estão dispostos fisicamente independentes da sua posição na estrutura lógica;

- os componentes se relacionam através de elos (apontadores);

LISTA SIMPLESMENTE ENCADEADA

Cada nodo da lista com encadeamento simples é formado por, no mínimo, 2 campos



Próximo é o campo que contém o endereço físico (relativo) do próximo nodo da lista.

Declaração de um nodo (registro) de um lista simp. encadeada:

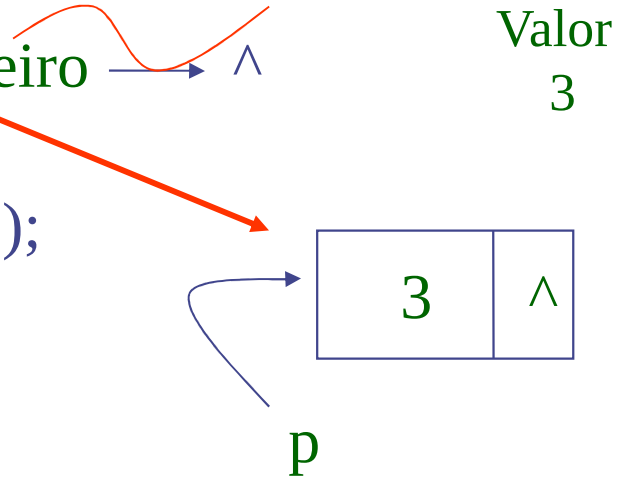
```
registro nodo
inicio
   inteiro dados;
   registro nodo *proximo;
fim
```

```
 Declaração em C
typedef struct nodo {
   int dados;
   struct nodo *proximo;
};
```

LISTA SIMPLESMENTE ENCADEADA

Primeiro e uma variável do tipo apontador que armazena o endereço do primeiro nodo da lista enc

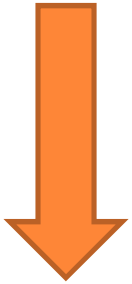
```
inicio
primeiro = nulo;
        Escrever ('Digite um valor: ');
        ler (valor);
        aloca (p);
        p->dados = valor;
        p->próximo = primeiro;
        primeiro = p;
 fim.
```



PARA ALOCAR... RELEMBRANDO

- aloca (p);

- **p = (struct nodo *) malloc (sizeof(struct nodo));**



Memory alloc – aloca memória para o tamanho da estrutura

CRIAR LISTA – DEFINIR STRUCT

```
typedef struct nodo {

   int dados;
   nodo *proximo;

} inicio, fim;
```

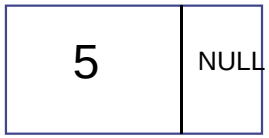
CRIAR LISTA – ENDEREÇO DO NÓ - VALOR 5 – VALOR DO PRÓXIMO

```
int main(){

nodo *fim=NULL;
nodo *inicio= NULL;

inicio = (struct nodo *) malloc (sizeof(struct nodo));
if (inicio)
{
        inicio->dados = 5;
        inicio->proximo = NULL;
        fim = inicio;

        printf("Ponteiro:%d\t valor:%d\t proximo:%d\n",inicio,inicio->dados,inicio->proximo);


         system("pause");
        return 0;
        }
}
```

```
#include <stdio.h>
#include <stdlib.h> //getche
#include <conio.h>
#include <windows.h> //gotoXY
#include <ctype.h> //toupper
#define MAX 5

typedef struct nodo {

      int dados;
      nodo *proximo;

} inicio, fim;
int main(){
nodo *fim=NULL;
nodo *inicio= NULL;
inicio = (struct nodo *) malloc (sizeof(struct no
if (inicio)
{
          inicio->dados = 5;
          inicio->proximo = NULL;
          fim = inicio;
           printf("Ponteiro:%d\t valor:%d\t proximo:%d\n",inicio,inicio->dados,inicio->proximo);
          system("pause");
          return 0;
          }
}
```

Estrutura NODO

Aloca memória



CRIANDO A FUNÇÃO - INSERIR LISTA

```
int insere_direita(struct no **inicio, struct no **fim, int valor){
        struct no *p;
        p = (struct no *) malloc(sizeof(struct no));
        if(p){
                p->dados = valor;
                p->proximo = NULL;
                if(*inicio == NULL)
                        *inicio = p;
                else
                        (*fim)->proximo = p;
                        *fim = p;
        }
```

CRIANDO A FUNÇÃO ... CRIA LISTA

```
/Função
  //Cria lista
   int cria_lista(struct no **inicio, struct no **fim){
         *inicio = NULL;
         *fim = NULL;
   }
```

CRIANDO A FUNÇÃO ... IMPRIME

```
//Imprime lista
    int mostra_lista(struct no *inicio){
            struct no *prox;
            prox = inicio;
            while(prox != NULL){
                    printf("%i = %i -> %i\n", prox, prox->dados, prox->proximo);
                    prox = prox->proximo;
            }
    }
```

EXERCÍCIOS

- Faça um código em C para criar uma lista encadeada, permitindo ao usuário digitar um valor, e ao término você deve mostrar para ele o resultado:

- Endereço Valor Próximo

- Primeiro faça para um NO.
- Depois faça para 2 nós
- Depois faça para 3 nós
- Isso fica bom??? Qual a melhor alternativa?
  - OBS: tente trabalhar com funções
  - OBS: faça um menu: inserir / imprimir / sair