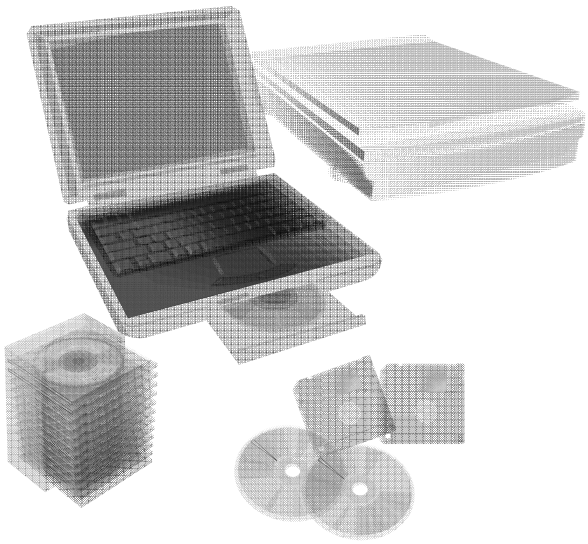
Introdução à Computação II



Listas dinâmicas

Profa.: ***Adriane Beatriz de Souza Serapião***

adriane@rc.unesp.br

Listas dinâmicas

- **Definição: é uma lista encadeada de pares, onde cada par é representado por um registro, constituído por: (elemento, apontador).**

- **<u>Desvantagens</u>:**
  - **exige mais espaço (existe um apontador adicional por elemento).**
  - **não é possível acessar um elemento diretamente.**
  - **acessar um elemento exige um caminhamento na lista na ordem exibida pelos elementos (leitura linear seguindo apontadores).**
- **<u>Vantagens</u>:**
  - **Operações de inserção e remoção de elementos.**

Listas dinâmicas

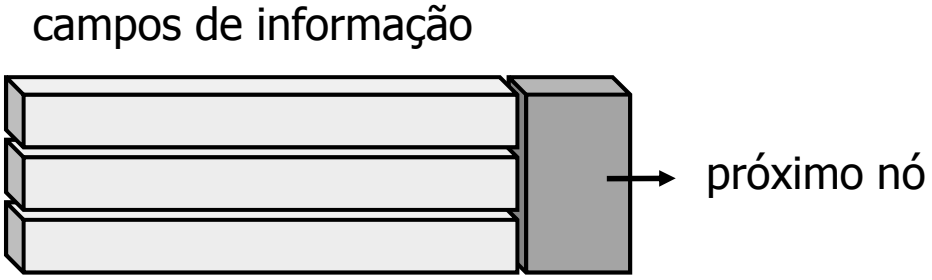
- **Ponteiros**
  - **Estruturas de dados dinâmicas: estruturas de dados que contém ponteiros para si próprias.**

```
struct lista {
  char nome_tarefa[30];
  float duracao;
  char responsavel[30];
  ...
  lista *prox;
};
```

ponteiro para a própria estrutura lista

Listas dinâmicas

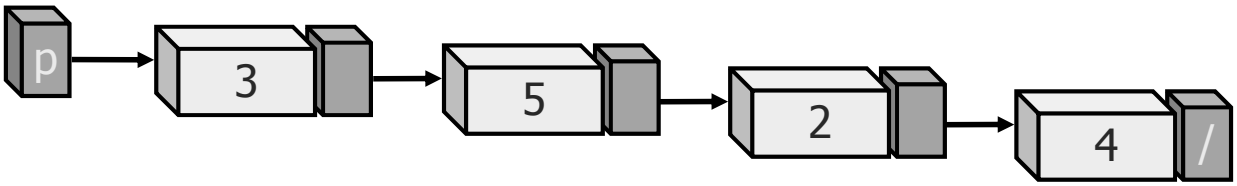
- **Representação gráfica de um elemento da lista:**



  - **Cada item é encadeado com o seguinte, mediante uma variável do tipo ponteiro.**
  - **Permite utilizar posições não contíguas de memória.**
  - **É possível inserir e retirar elementos sem necessidade de deslocar os itens seguintes da lista.**

Listas dinâmicas

- Cada item em particular de uma lista pode ser chamado de elemento, nó, célula, ou item.
- O apontador para o início da lista também é tratado como se fosse uma célula (cabeça), para simplificar as operações sobre a lista.
- O símbolo / representa o ponteiro nulo (NULL), indicando o fim da lista.



Operações sobre listas dinâmicas encadeadas

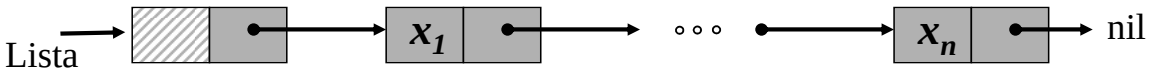
- Podemos realizar algumas operações sobre uma lista encadeadas, tais como:
  - Inserir itens;
  - Retirar itens;
  - Buscar itens.
- Para manter a lista ordenada, após realizar alguma dessas operações, será necessário apenas movimentar alguns ponteiros (de um a três elementos).

Operações sobre listas dinâmicas

- **Outras operações possíveis:**
  - **Criar uma lista**
  - **Destruir uma lista**
  - **Ordenar uma lista**
  - **Intercalar duas listas**
  - **Concatenar duas listas**
  - **Dividir uma lista em duas**
  - **Copiar uma lista em outra**

Implementação de listas dinâmicas

- **Há uma** célula cabeça **para simplificar as operações sobre a lista.**

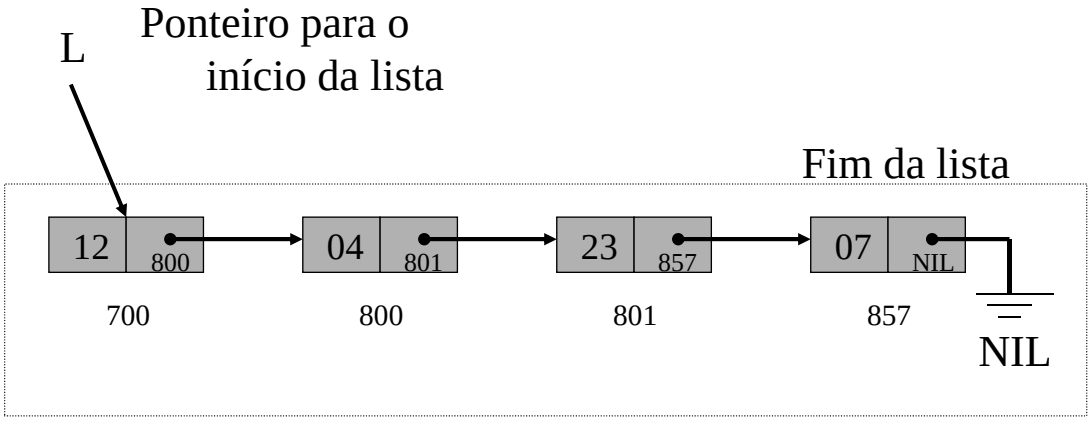


Estrutura da lista dinâmica

- **A lista é constituída de células.**
- **Cada célula contém um item da lista e um ponteiro para a célula seguinte.**
- **O registro TipoLista contém um ponteiro para a célula cabeça e um ponteiro para a última célula da lista.**

Estrutura da lista dinâmica

- **Lista Encadeada**



Lista encadeada

Estrutura da lista dinâmica

```
typedef struct {
     int Item;
     Apontador Prox;
} Celula;

typedef struct Celula Apontador;
```


Operações sobre listas dinâmicas

```
void FLVazia (Apontador *Lista)
{
   Lista     =     (Apontador     *)     malloc(
       sizeof(Apontador));
    Lista->Prox = NULL;
};
```

```
int Vazia (Apontador *Lista)
{
   return (Lista->Prox == NULL);
};
```

Operações sobre listas dinâmicas

```
void Insere_fim (TipoItem x, Apontador *Lista)
{
   Apontador Novo, p;
   Novo = (Apontador *) malloc(sizeof(Apontador));
   Novo->Item = x;
   Novo->Prox = NULL;
   p = Lista->Prox;
   while (p->Prox != NULL) p = p->Prox;
   p->Prox = Novo;
};

void Insere_inicio (TipoItem x, Apontador *Lista)
{
   Apontador p;
   p = (Apontador *) malloc(sizeof(Apontador));
   p->Item = x;
   p->Prox = Lista->Prox;
   Lista->Prox = p;
};
```


Operações sobre listas dinâmicas

```
void Retira (Apontador p, Apontador *Lista,
   TipoItem *Item)
{--O ITEM A SER RETIRADO É O SEGUINTE AO APONTADO POR P --}
{
   Apontador q;
   if (Vazia(*Lista) || (p == NULL) || (p->Prox
   == NULL))
   {
      printf( "Erro : Lista vazia ou posicao
              nao existe\n" );
      return;
   }
   q = p->Prox;
   *Item = q->Item;
    p->Prox = q->Prox;
   free(q);
};
```

Operações sobre listas dinâmicas

```
void Imprime (Apontador Lista)
{
   Apontador Aux;
   Aux = Lista->Prox;
   while (Aux != NULL)
   {
       printf("%d12\n",Aux->Item.Chave);
       Aux = Aux->Prox;
   };
};
```


Listas dinâmicas – vantagens e desvantagens

- **Vantagens:**
  - **Permite inserir ou retirar itens do meio da lista a um custo constante (importante quando a lista tem de ser mantida em ordem).**
  - **Bom para aplicações em que não existe previsão sobre o crescimento da lista (o tamanho máximo da lista não precisa ser definido *a priori*).**
- **Desvantagem:**
  - **utilização de memória extra para armazenar os ponteiros.**
- **Quando usar:**
  - **quando for possível fazer uma boa previsão do espaço utilizado (lista principal + lista auxiliar) e quando o ganho dos movimentos sobre a perda do acesso direto a cada elemento for compensador.**

Atenção!

- **Há diversas maneiras de construir a estrutura de uma lista dinâmica.**
- **Aqui foi apresentada uma lista que possui apontadores para o início e o fim da lista.**
- **Estude também a versão onde há somente um único apontador para o início da lista.**

Exercícios – listas

- **Faça um programa para inverter uma lista:**

```
void InverteLista(Apontador *p)
{
  Apontador q, r, s;
  q = NULL;
  r = p;
  while (r != NULL) {
    s = r->prox;
    r->prox = q;
    q = r;
    r = s
  };
  p = q
};
```

Exercícios – listas

- **Implemente algumas das operações básicas com listas dinâmicas que não foram apresentadas aqui:**
  - **Inserir elemento em uma posição qualquer da lista;**
  - **Retirar elemento no início da lista;**
  - **Retirar elemento no fim da lista;**
  - **Retirar elemento de uma posição qualquer da lista;**
  - **Buscar um elemento qualquer dentro de uma lista;**
  - **Dada uma posição qualquer em uma lista, acessar seu conteúdo;**
  - **Concatenar listas;**
  - **Copiar uma lista em outra.**

Bibliografia

- **ZIVIANI, N. *Projeto de Algoritmos com Implementações em Pascal e C*, (4a. ed.). Thomson, 2011.**