Estruturas de Dados

Engenharia Informática

1º Ano - 2º Semestre

Francisco Morgado
Escola Superior de Tecnologia e Gestão de Viseu

3. ESTRUTURAS DINÂMICAS

3.1 Stacks (Pilhas)

3.2 Filas de espera

3.3 Listas ligadas ordenadas

3.4 Listas bi-ligadas ordenadas

3.5 Hashing

3.6 Árvores binárias

3.4 Listas bi-ligadas ordenadas

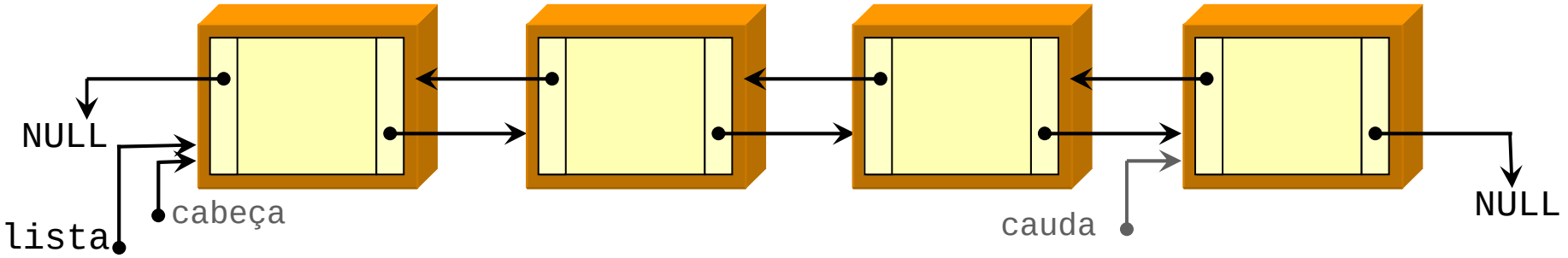
Conceito

Inserção de elementos

Remoção de elementos

Listagem de elementos

Listas Bi-Ligadas Ordenadas - Conceito




```
typedef struct
{
    char nome[80];
    char bi[12];
    int  numero; /*chave*/
}Informacao;
```

```
typedef struct nodo
 {
  Informacao dados;
  struct nodo *anterior;
  struct nodo *seguinte;
 }Nodo;
```

```
typedef Nodo* ListaDupla;
```

```
// criar lista vazia
ListaDupla lista = NULL;
```

- **Operações na Lista**
  - Inserir elemento na ordem correta
  - Retirar qualquer elemento
  - Listar elementos (ordenadamente)
  - Listar elementos por ordem inversa

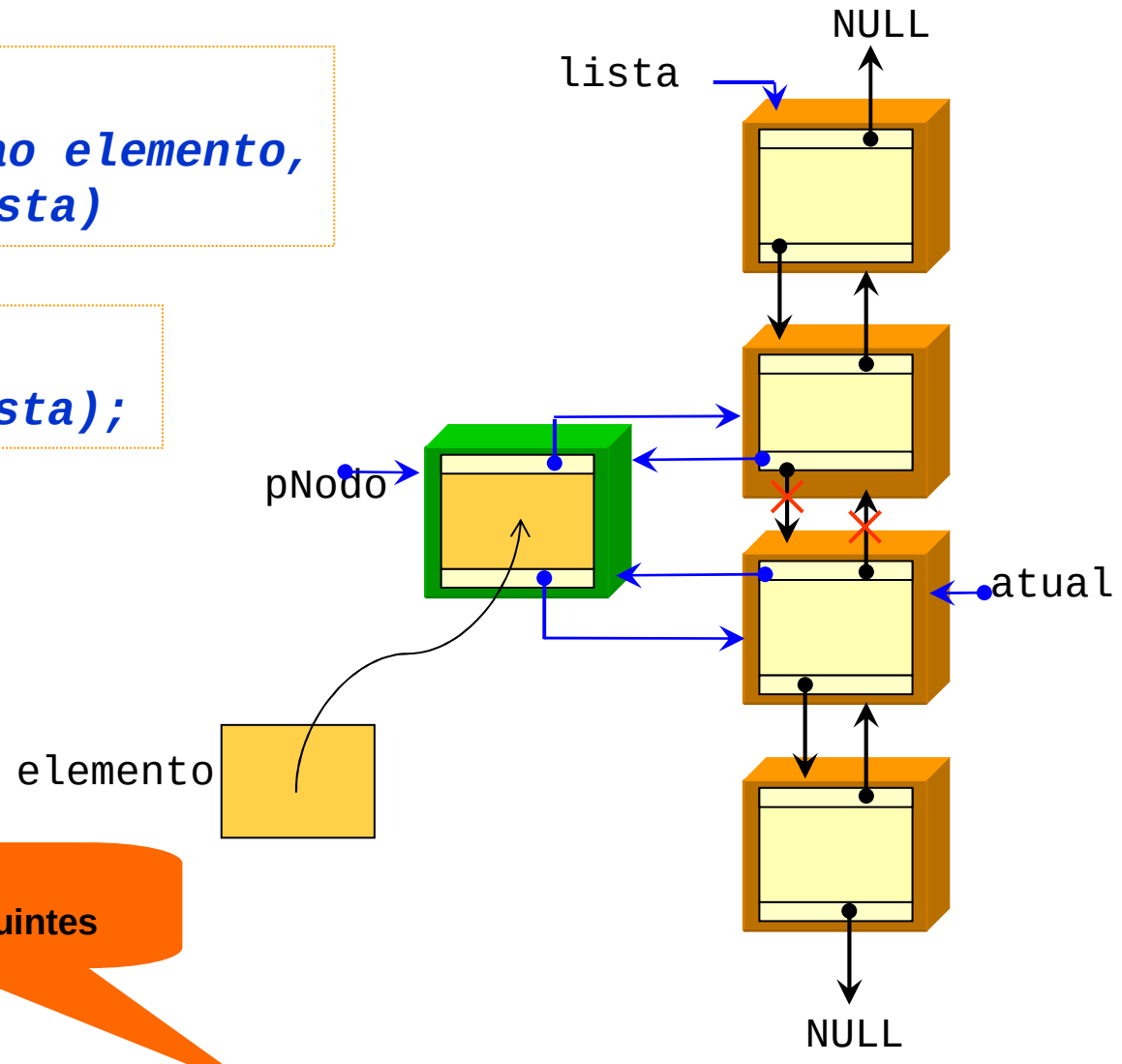
Listas Bi-Ligadas Ordenadas - Inserção de elementos

Cabeçalho

```
void inserir(Informacao elemento,
             ListaDupla *lista)
```

Chamada

```
inserir(elemento, &lista);
```



FUNÇÃO:
Diapositivos seguintes

```
void inserir(Informacao elemento, ListaDupla *lista)
{
  int av = 1;
  Nodo* atual;
  Nodo* pNodo = (Nodo*) malloc(sizeof(Nodo));
  pNodo->dados = elemento;
  if (*lista == NULL)      // se lista vazia
  {
      pNodo->seguinte = pNodo->anterior = NULL;
      *lista = pNodo;
  }
  else           // se lista não vazia
  {
    atual = *lista;
    while(av==1)  // procura posição
        if(pNodo->dados.numero < atual->dados.numero)
            av=2; //inserir antes de atual
        else if (atual->seguinte == NULL)
            av=0; // atual é último: inserir no fim
        else
            atual = atual->seguinte; // seguinte...
```

```
    // continuação
    if(atual == *lista && av==2) { //inserir no início
        pNodo->anterior = NULL;
        pNodo->seguinte = atual;
        atual->anterior = pNodo;
        *lista = pNodo;
    }
    else if(av==2) { //inserir no meio
        pNodo->anterior = atual->anterior;
        pNodo->seguinte = atual;
        (atual->anterior)->seguinte = pNodo;
        atual->anterior = pNodo;
    }
    else {       // av == 0 então inserir no fim
        pNodo->anterior = atual;
        pNodo->seguinte = NULL;
        atual->seguinte = pNodo;
    }
  } //FIM do else 'se lista não vazia'
}
```

```
Nodo* remover(int chave, ListaDupla *lista)
{
      Nodo* ret,  *act;  int av=1;
      if (*lista == NULL)   //se lista vazia
            return NULL;
      act = *lista; // lista não vazia
      while(av == 1)
            if(act == NULL)
                  av = 0;
            else if(act->dados.numero == chave)
                  av = 2;
            else
                  act = act->seguinte;
      // ciclo while só termina quando encontra o
      // elemento pretendido ou chega ao fim da lista
```

Chamada

```
pNodo = remover(nAluno, &lista);
```

```
    if(act)      // ou if (av == 2)
    {     // se o elemento existe
        ret=act;
        if (*lista==act) //remoção no início
        {
            *lista=act->seguinte;
            (*lista)->anterior = NULL;
        }
        else  //remoção no meio
        {
          (act->anterior)->seguinte = act->seguinte;
          if (act->seguinte)    //se não é o último
            (act->seguinte)->anterior =act->anterior;
        }
    }
    else // não existe
        ret = NULL;
    return ret;
}
```

Contar e mostrar elementos numa lista bi-ligada

```
void mostrar(ListaDupla lista)
{
      while(lista != NULL) {
            mostrar(lista->dados);
            lista = lista->seguinte;
      }
}
```

Chamadas

Exercício…

```
void mostrarInverso(ListaDupla lista)
{
    Nodo* pNodo = lista;
    if (lista == NULL)
        printf("\nLista vazia\n");
    else {   // percorre do primeiro nodo até ao último
        while(pNodo->seguinte != NULL)
            pNodo = pNodo->seguinte;
        do {// percorre do último nodo até ao primeiro, mostrando
            mostrar(pNodo->dados);
            pNodo = pNodo->anterior;
        }while(pNodo != NULL);
    }
}
```

Contar elementos numa lista bi-ligada. Função Menu ( )

```
int NumElementos(ListaDupla lista) {
    int n = 0;
    while (lista != NULL) {
        n++;
        lista = lista->seguinte;
    }
    return n;
}
```

Função main()

Exercício…

```
char menu()  {
  char op; fflush(stdin);
  printf("\n\n   |-----------------------------------------|");
  printf("\n   |   1 - Inserir elemento                  |");
  printf("\n   |   2 - Retirar elemento                  |");
  printf("\n   |   3 - Mostrar elementos ordem crescente |");
  printf("\n   |   4 - Mostrar elementos ordem decrescente |");
  printf("\n   |   5 - Comprimento da Lista              |");
  printf("\n   |   0 - Sair                              |");
  printf("\n   |-----------------------------------------|\n");
  do {
    printf("\n  Qual a sua opcao ? "); op=getchar();
  } while (op < '0' || op > '5');
  return op;
}
```