UEM/CTC – Departamento de Informática
Curso: Ciência da Computação
Professor: Flávio Rogério Uber

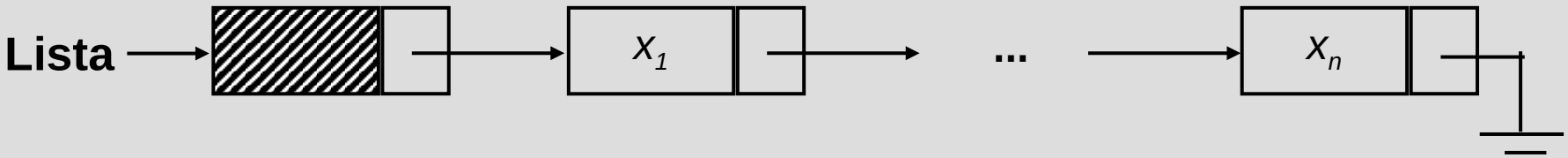
Estrutura de Dados
(6884/3 e 4)

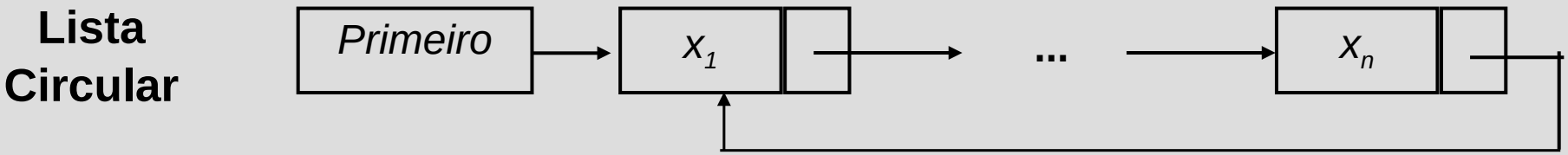
Listas Duplamente Encadeadas

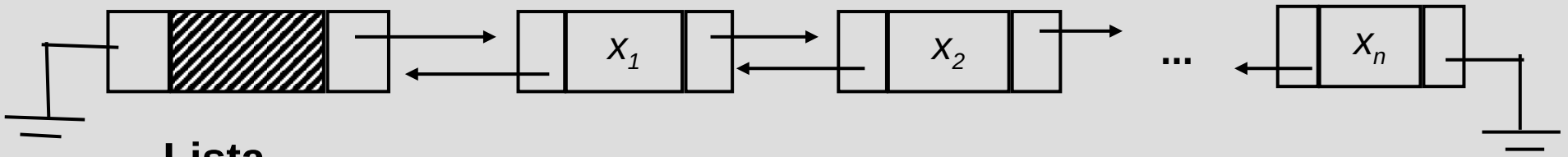
Listas Duplamente Encadeadas

- Lista encadeada onde cada nó tem uma ligação com o próximo nó e com o nó anterior
- Existe a identificação do primeiro e do último nó
- É possível ter a referência com o nó anterior sem a necessidade de percorrer a lista toda
- É possível se deslocar na lista em ambos os sentidos

Listas Duplamente Encadeadas







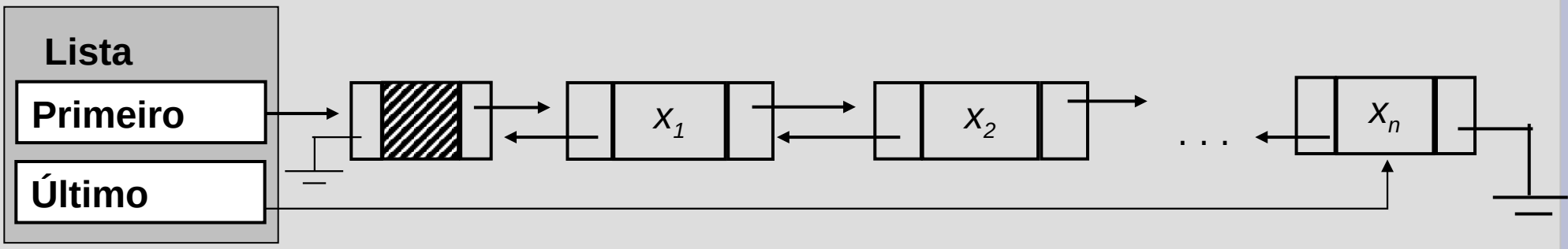
Lista Duplamente Encadeada

Listas Duplamente Encadeadas

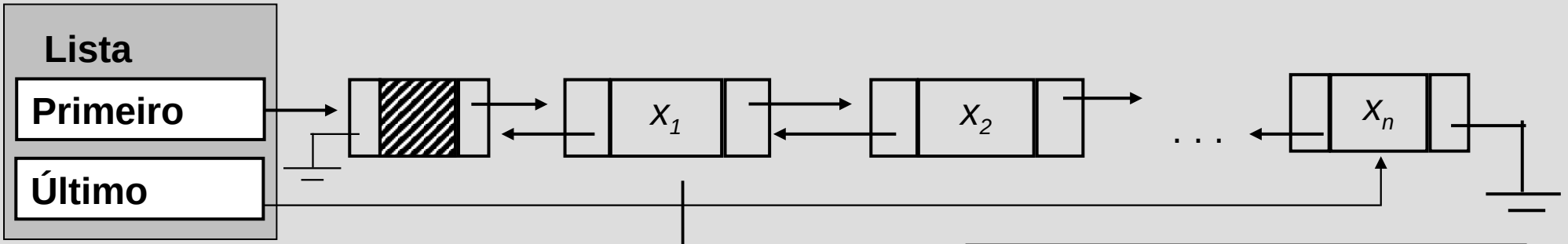
- Inserção/Remoção
  - Todo elemento tem uma ligação com o “próximo” e com o “anterior”
  - A inserção pode ocorrer no início, no meio e no final da lista
  - Idem para a remoção

Listas Duplamente Encadeadas

- Implementação
  - O registro TipoLista tem um apontador a primeira célula e outro para a última célula da lista;



Listas Duplamente Encadeadas



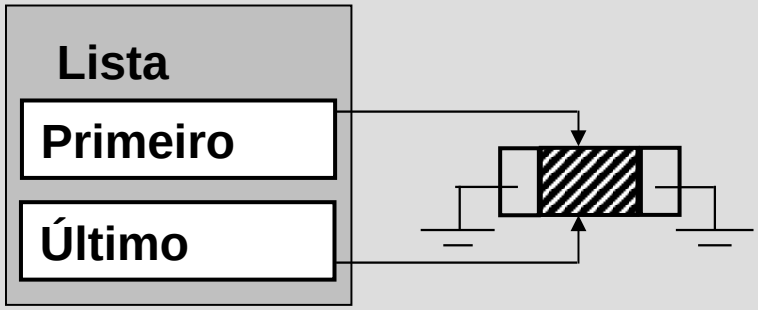

```
typedef struct
 {
  int chave;
  char nome [30];
  int idade;
  //outros campos
 }TipoItem;
```

```
typedef struct
 {
  Celula* Primeiro;
  Celula* Ultimo;
 }TipoLista;
```

```
struct Celula
 {
  TipoItem Item;
  Celula* Prox;
  Celula* Ant;
 };
```

Listas Duplamente Encadeadas

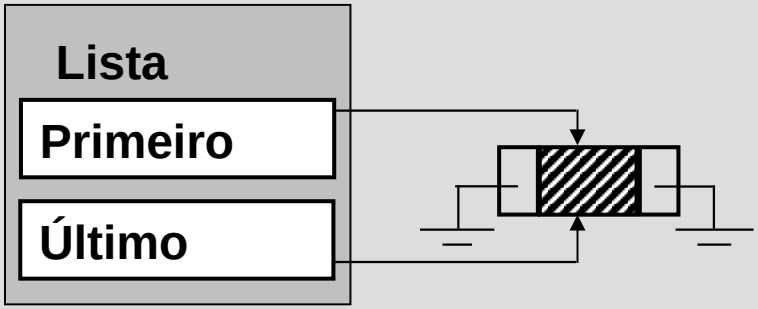
void FLVazia (TipoLista * Lista);




```
void FLVazia (TipoLista *Lista)
 {
  Lista -> Primeiro = (Celula*) malloc(sizeof(Celula));
  Lista -> Ultimo = Lista -> Primeiro;
  Lista -> Primeiro -> Prox = NULL;
  Lista -> Primeiro -> Ant = NULL;
 }
```

Listas Duplamente Encadeadas

int Vazia (TipoLista Lista);

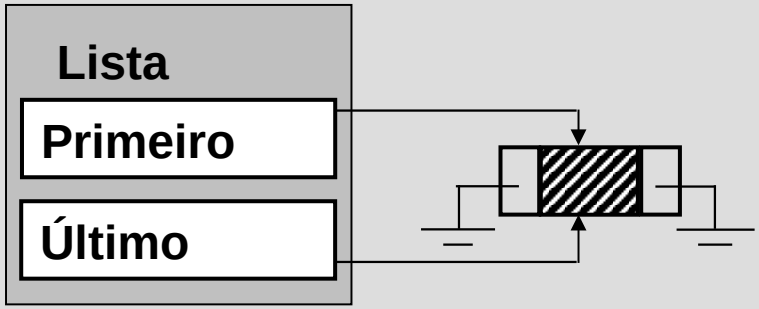


?

```
int Vazia (TipoLista Lista)
 {
  return (Lista.Primeiro == Lista.Ultimo);
 }
```

Listas Duplamente Encadeadas

void Insere (TipoItem x, TipoLista *Lista);



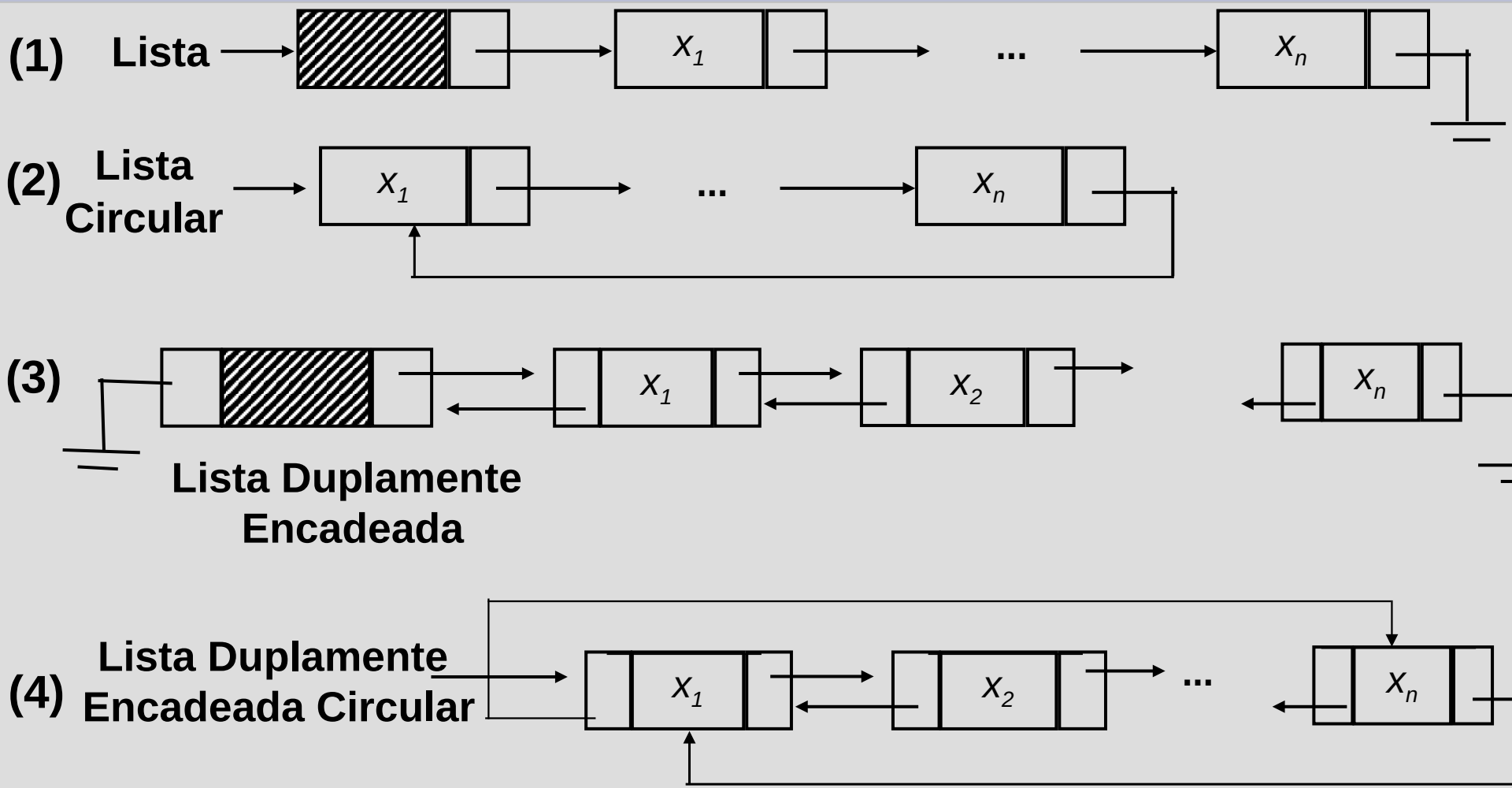

```
void Insere (TipoItem x, TipoLista *Lista)
 {
  Lista -> Ultimo -> Prox = (Celula*) malloc(sizeof(Celula));
  Lista -> Ultimo -> Prox -> Ant = Lista -> Ultimo;
  Lista -> Ultimo = Lista -> Ultimo -> Prox;
  Lista -> Ultimo -> Item = x;
  Lista -> Ultimo -> Prox = NULL;
 }
```

Listas Duplamente Encadeadas

```
void Imprime (TipoLista Lista)
 {
  Celula* Aux;
  Aux = Lista.Primeiro -> Prox;
  while (Aux != NULL)
   {
    printf ("\n\nCodigo do elemento: %d",Aux->Item.chave);
    ...
    Aux=Aux->Prox;
    i++;
   }
 }
```

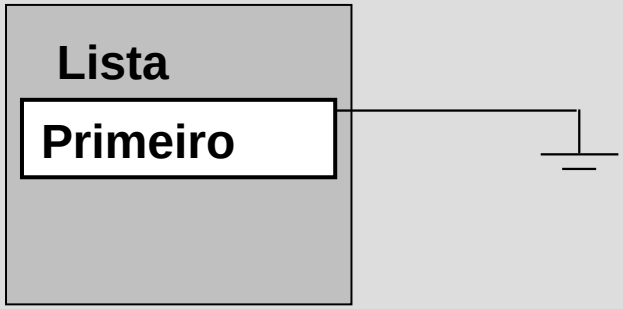
```
void ImprimeINV (TipoLista Lista)
 {
  Celula* Aux;
  Aux = Lista.Ultimo;
  while (Aux != Lista.Primeiro)
   {
    printf ("\n\nCodigo do elemento: %d", Aux->Item.chave);
    ...
    Aux=Aux->Ant;
    i++;
   }
 }
```

Listas Duplamente Encadeadas Circulares



Listas Duplamente Encadeadas Circulares

void Insere (TipoItem x, TipoLista *Lista);



Inserção em Lista Circular

```
void Insere (TipoItem x, TipoLista *Lista)
 {
  *Celula pins;
  *Celula paux;
  pins = (*Celula) malloc(sizeof(Celula));
  if (Lista->Primeiro == NULL)
  {
     Lista->Primeiro = pins;
     pins->Prox = pins;
  }
  else
  {
     paux = Lista->Primeiro;
     while (paux->Prox != Lista->Primeiro)
     {
        paux = paux->Prox;
     }
     pins->Prox = Lista->Primeiro;
     paux->Prox = pins;
  }
  pins->Item = x;
 }
```

```
void Insere (TipoItem x, TipoLista *Lista)
 {
  *Celula pins;

  pins = (*Celula) malloc(sizeof(Celula));
  if (Lista->Primeiro == NULL)
  {
     Lista->Primeiro = pins;
     pins->Prox = pins;
     pins->Ant = pins;
  }
  else
  {
     pins->Prox = Lista->Primeiro;
     pins->Ant = Lista->Primeiro->Ant;
     pins->Ant->Prox = pins;
     pins->Prox->Ant = pins;
  }
  pins->Item = x;
 }
```

Inserção em Lista Duplamente Encadeada Circular