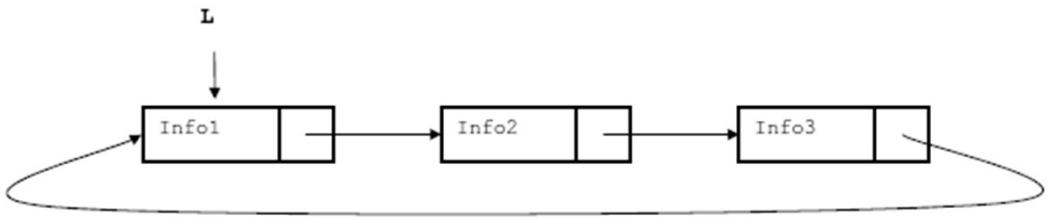
Listas Circulares
e
Listas Duplamente encadeadas

Listas circular

- Lista circular
- O último elemento tem como próximo o primeiro elemento da lista, formando um ciclo.
- A lista pode ser representada por um ponteiro para um elemento inicial qualquer da lista.



Listas circular

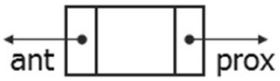
```
/* função imprime: imprime valores dos elementos */
void imprimeCircular (Lista* l)
{
  Lista *p = l; /* faz p apontar para o nó inicial */
  /* testa se lista não é vazia e então percorre com do-while
*/
  if (p) do
  {
     printf("%d\n", p->info); /* imprime informação do nó */
     p = p->prox; /* avança para o próximo nó */
  } while (p != l);
}
```

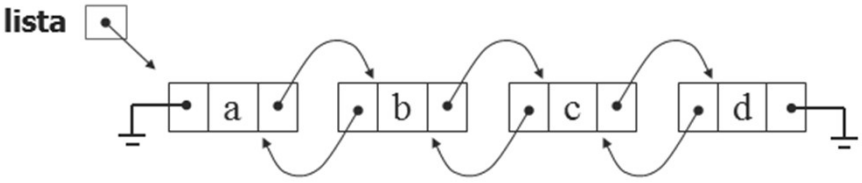
Listas circular

```
/* inserção no início: retorna a lista atualizada */
Lista *insereLista (Lista *l, int i)
{
Lista *aux;
Lista *novo = (Lista*) malloc(sizeof(Lista));
novo->info = i;
novo->prox = l;
if (l==NULL)
    novo->prox = novo;
else
{
    aux=l;
    do {
        aux=aux->prox;
    } while (aux->prox!=l);
   aux->prox = novo;
}
return novo;
}
```

Lista duplamente encadeada

- Cada nó possui dois ponteiros: um para o elemento anterior e outro para o próximo elemento (ant e prox)





Página 5

Lista duplamente encadeada

```
/* inserção no início: retorna a lista atualizada */
Lista *insereLista (Lista *l, int i)
{
Lista *aux;
Lista *novo = (Lista*) malloc(sizeof(Lista));
novo->info = i;
novo->prox = l;
novo->ant = NULL;
if(l!=NULL)
    l->ant = novo;
return novo;
}
```

Lista duplamente encadeada

```
void imprimeInverso (Lista *l)
{
Lista *p,*a;
for (p = l; p->prox != NULL; p = p->prox);
printf("\nImpressao da lista invertida: \n");
for (a = p; a != NULL; a = a->ant)
   printf("info = %d\n", a->info);
   printf("================== fim da lista inversa");
}
```

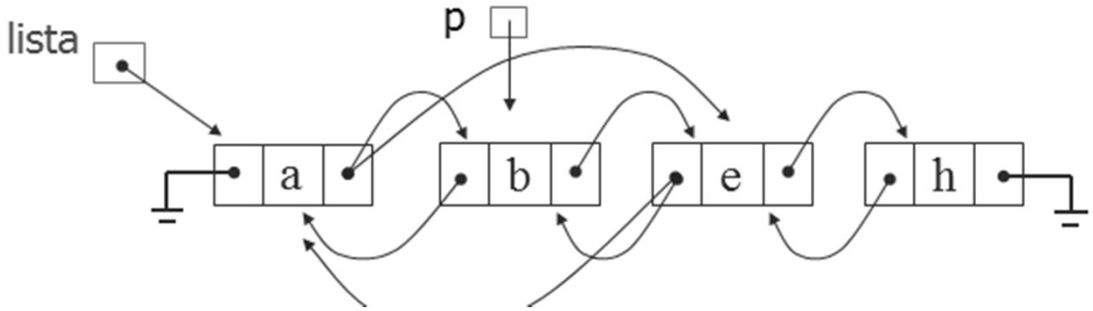
Lista duplamente encadeada

- A remoção é mais trabalhosa, pois é preciso acertar a cadeia nos dois sentidos
- Em compensação, pode-se retirar um elemento conhecendo-se apenas o ponteiro para ele
- Utiliza-se uma função de busca para localizar o elemento e em seguida o encadeamento é ajustado
- Ao final, o elemento é liberado
- Sendo **p** o ponteiro para o elemento a ser excluído, se o elemento estiver no meio da lista, devemos fazer:

```
p->ant->prox = p->prox;
p->prox->ant = p->ant;
```

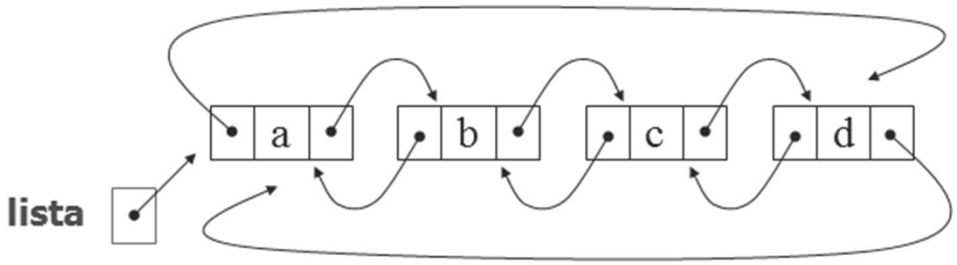
Lista duplamente encadeada

- Caso o elemento esteja em um extremo da lista, existem outras condições:
  - se p for o primeiro, não se pode referenciar p->ant, pois ele é NULL; o mesmo acontece para p->prox quando é o último
  - além disso, se for o primeiro, é preciso atualizar o ponteiro da lista



Lista duplamente encadeada circular

- Cada nó possui dois ponteiros: um para o elemento anterior e outro para o próximo elemento (ant e prox)
- O anterior do primeiro é o último e o próximo do último é o primeiro



Página 10