UNISINOS - UNIVERSIDADE DO VALE DO RIO DOS SINOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS (C6/6) – Curso: Informática

LABORATÓRIO II – AULA 08

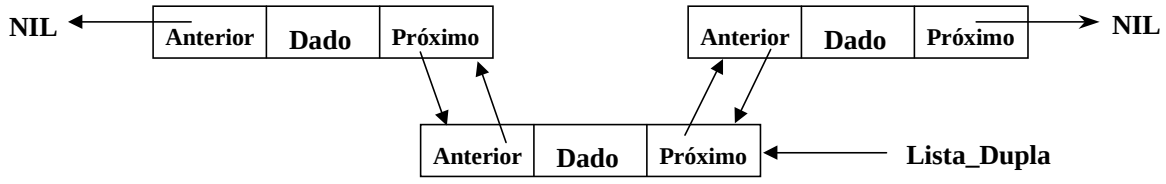
Disciplina: Linguagem de Programação PASCAL
Professor responsável: *Fernando Santos Osório*
Semestre: 99/2
Horário: 21 e 41

E-mail: *osorio@exatas.unisinos.br*
Web:
http://www.inf.unisinos.br/~osorio/lab2.html
Xerox : *Pasta 54* – LAB. II (Xerox do C6/6)

LISTAS DUPLAMENTE ENCADEADAS
Alocação Dinâmica de Memória

As listas duplamente encadeadas são similares as listas simplesmente encadeadas, mas com a diferença de que nestas cada nodo além de possuir um ponteiro para o próximo nodo da lista, possuem também um ponteiro para o nodo anterior. Desta forma podemos percorrer uma lista duplamente encadeada nos dois sentidos, sendo que não somos mais "obrigados" a guardar um ponteiro para o início da lista. Se tivermos um ponteiro indicando um nodo qualquer da lista duplamente encadeada, sempre será possível chegar ao seu nodo inicial e também ao seu nodo final.

A figura abaixo mostra um exemplo de lista duplamente encadeada:



Usualmente teremos a seguinte estrutura de dados:

```
Type
    Tipo_Dado          = Integer;
    Ptr_Nodo_LD        = ^Nodo_Lista_Dupla;
    Nodo_Lista_Dupla   = Record
                             Dado: Tipo_Dado;
                             Anterior, Proximo : Ptr_Nodo_LD;
                         End;

Var
    Lista_Dupla: Ptr_Nodo_LD;
    Novo: Ptr_Nodo_LD;

{ Criando uma lista duplamente encadeada com 3 nodos apenas }
{  Os nodos são inseridos sempre no final da lista neste exemplo }
```

```
Begin
    New (Novo);                    { Alocação dinâmica: cria o primeiro nodo     }
    Novo^.Dado := 1;               {  Coloca o dado no nodo que foi alocado      }
    Novo^.Anterior := NIL;         {  Inicialmente o nodo não tem anterior...    }
    Novo^.Proximo := NIL;          {  e também não tem próximo.                  }
    Lista_Dupla:= Novo;            { Lista_Dupla aponta para o 1º nodo da lista  }

    New(Novo);                     { Alocação dinâmica: cria o segundo nodo      }
    Novo^.Dado := 2;               { Coloca o dado no nodo que foi alocado       }
    Lista_Dupla^.Proximo:=Novo;    { O próximo depois do Lista_Dupla é o novo nodo }
    Novo^.Anterior:=Lista_Dupla;   { O anterior do novo nodo é o Lista_Dupla     }
    Novo^.Proximo := NIL;          { O novo nodo ainda não possui próximo        }
    Lista_Dupla := Novo;           {  Lista_Dupla aponta para o fim da lista     }

    New(Novo);                     { Alocação dinâmica: cria o terceiro nodo     }
    Novo^.Dado := 3;               { Coloca o dado no nodo que foi alocado       }
    Lista_Dupla^.Proximo:=Novo;    { O próximo depois do Lista_Dupla é o novo nodo }
    Novo^.Anterior:=Lista_Dupla;   { O anterior do novo nodo é o Lista_Dupla     }
    Novo^.Proximo := NIL;          { O novo nodo ainda não possui próximo        }
    Lista_Dupla := Novo;           {  Lista_Dupla aponta para o fim da lista     }
End.
```

As listas duplamente encadeadas permitem que se percorra a lista nos dois sentidos, o que pode ser bastante interessante de acordo com o problema a ser tratado... entretanto acabamos usando mais memória, pois para cada nodo de informação teremos que ter associados dois ponteiros (anterior e próximo). Sempre que formos construir uma estrutura de dados devemos avaliar itens como:

- *Espaço em memória ocupado pela estrutura de dados adotada;*
- *Tempo gasto para acessar um dado qualquer na estrutura dados adotada;*
- *Tempo gasto para alterar a estrutura de dados adotada: inserção e remoção.*

EXERCÍCIOS – *Listas duplamente encadeadas*

1. Faça um programa para a manipulação de listas duplamente encadeadas com alocação dinâmica na memória do computador. Crie as seguintes rotinas genéricas de manipulação de dados:

   Definições elementares:

```
Type
    Tipo_Dado          =  Integer;
    Ptr_Nodo_LD        =  ^Nodo_Lista_Dupla;
    Nodo_Lista_Dupla   =  Record
                             Dado: Tipo_Dado;
                             Anterior, Proximo : Ptr_Nodo_LD;
                          End;
```

Rotinas:

Procedure Inicializa_LD (Var N: Ptr_Nodo_LD);

Procedure Posiciona_LD_Inicio (Var N: Ptr_Nodo_LD); {N = Um Nodo qualquer da Lista Dupla }

Procedure Posiciona_LD_Final (Var N: Ptr_Nodo_LD);

Procedure Insere_LD_Antes (Var N: Ptr_Nodo_LD; Dado: Tipo_Dado);

Procedure Insere_LD_Depois (Var N: Ptr_Nodo_LD; Dado: Tipo_Dado);

Procedure Insere_LD_Ordenando (Var N: Ptr_Nodo_LD; Dado: Tipo_Dado);

Function Percorre_LD (Var N: Ptr_Nodo_LD; Var Dado: Tipo_Dado): Boolean; {Avança um }

Function Quantidade_LD (N: Ptr_nodo_LD): Integer; { Indica quantos dados tem na lista }

Procedure Exibe_LD (N: Ptr_Nodo_LD); { Exibe os dados da lista }

Function Pesquisa_LD (Var N: Ptr_Nodo_LD; Dado: Tipo_Dado): Boolean; { Procura dado }

Function Remove_LD_Nodo (Var N: Ptr_Nodo_LD; Var Dado:Tipo_Dado): Boolean;

Function Remove_LD_Dado (Var N: Ptr_Nodo_LD; Dado: Tipo_Dado): Boolean;

Procedure Apaga_LD (Var N: Ptr_Nodo_LD); { Apaga todo o conteúdo da lista }

2. Implemente as rotinas: *inicializa_pilha, insere_pilha, retira_pilha,* e *exibe_pilha*, usando as rotinas de manipulação de listas duplamente encadeadas que foram implementadas no exercício anterior.

3. Implemente as rotinas: *inicializa_fila, insere_fila, retira_fila,* e *exibe_fila*, usando as rotinas de manipulação de listas duplamente encadeadas que foram implementadas no exercício número 1.

4. Faça uma análise crítica das rotinas de manipulação de listas duplamente encadeadas que foram implementadas, no que se refere a sua utilização como rotinas de base para a implementação de filas. Considere os seguintes aspectos: inserção, remoção e quantidade de espaço alocado, sugerindo possíveis melhorias nestas rotinas.