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

LABORATÓRIO II – AULA 10

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

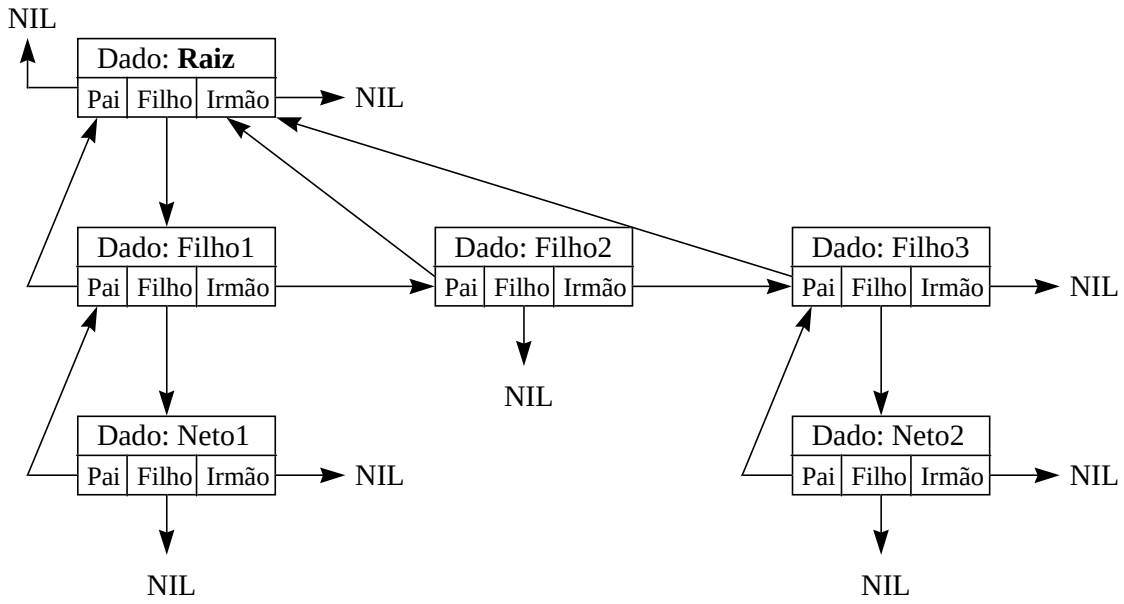
E-mail: ***osorio@exatas.unisinos.br***
Web:
http://www.inf.unisinos.br/~osorio/lab2.html
Xerox : *Pasta 54* – LAB. II (Xerox do C6/6)

ÁRVORES GENÉRICAS
Alocação Dinâmica de Memória

As árvores são estruturas de dados criadas usualmente através do uso de alocação dinâmica de memória, baseadas em listas encadeadas, que possuem um nodo superior (raiz / pai), apontando para os seus nodos filhos (folhas / filho). Por sua vez cada nodo pai pode possuir nodos filhos, e assim chegamos a definição "recursiva" de uma árvore: um nodo pai aponta para nodos filhos, onde estes nodos filhos também podem ser nodos pais.

Uma árvore genérica é um tipo especial de árvore onde podemos ter um número variável de nodos filhos associados a um nodo pai. Um exemplo de implementação de uma árvore genérica pode ser dado por uma estrutura onde cada nodo possui um ponteiro para o seu nodo pai, um ponteiro para o seu nodo filho, e um ponteiro para o seu nodo irmão. Assim sendo, cada nodo pode possuir um número ilimitado de filhos, pois o seu nodo filho aponta para uma lista ilimitada de nodos no mesmo nível na hierarquia da árvores (irmãos). Além disto, continuamos tendo uma estrutura em árvore, pois cada nodo possui seu nodo pai e seu(s) nodo filho(s).

A figura abaixo mostra um exemplo de uma árvore genérica:



Podemos ter a seguinte estrutura de dados para uma ***árvore genérica***:

```
Type
    Tipo_Dado          =  Integer;
    Ptr_Nodo_AG        =  ^Nodo_Arv_Gen;
    Nodo_Arv_Gen       =  Record
                             Dado: Tipo_Dado;
                             Pai, Filho, Irmao:  Ptr_Nodo_AG;
                          End;
Var
    Arvore_Generica: Ptr_Nodo_AG;
    Novo_Nodo      : Ptr_Nodo_AG;
    Ultimo_Filho   : Ptr_Nodo_AG;

{ Criando uma árvore genérica com 4 nodos apenas...  divididos em 2 níveis      }
{  2 níveis:  1º. nível – nodo raiz/pai, 2º. nível – 3 nodos filhos              }

Begin
    New (Novo_Nodo);              { Alocação dinâmica: cria o nodo raiz (pai)     }
    Novo_Nodo^.Dado := 1;         {  Coloca o dado no nodo que foi alocado        }
    Novo_Nodo^.Pai := NIL;        { O nodo raiz não tem pai (sem nodos acima)     }
    Novo_Nodo^.Irmao := NIL;      { Não possui nodos irmãos no mesmo nível        }
    Novo_Nodo^.Filho := NIL;      { Inicialmente não possui nodos filhos          }
    Arvore_Generica := Novo_Nodo;  { Arvore_generica aponta para a Raiz           }

    New(Novo_Nodo);               { Alocação dinâmica: cria o nodo filho1         }
    Novo_Nodo^.Dado := 11;        { Coloca o dado no nodo que foi alocado         }
    Novo_Nodo^.Pai :=Arvore_Generica;     { O pai do novo nodo é o nodo raiz      }
    Novo_Nodo^.Filho := NIL;      {  Inicialmente o nodo não tem filho            }
    Novo_Nodo^.Irmao := NIL;      {  e também não tem irmão                       }
    Arvore_Generica^.Filho := Novo_Nodo;  { O filho da raiz é o novo nodo         }
    Ultimo_Filho := Novo_Nodo;    { Indica que este foi o último filho inserido   }

    New(Novo_Nodo);               { Alocação dinâmica: cria o filho2 (irmão)      }
    Novo_Nodo^.Dado := 12;        { Coloca o dado no nodo que foi alocado         }
    Novo_Nodo^.Pai :=Arvore_Generica;     { O pai do novo nodo é o nodo raiz      }
    Novo_Nodo^.Filho := NIL;      {  Inicialmente o nodo não tem filho            }
    Novo_Nodo^.Irmao := NIL;      {  e também não tem irmão (mais moço)           }
    Ultimo_Filho^.Irmao := Novo_Nodo;     { O irmão aponta p/ novo irmão          }
    Ultimo_Filho := Novo_Nodo;    { Indica que este foi o último filho inserido   }

    New(Novo_Nodo);               { Alocação dinâmica: cria o filho2 (irmão)      }
    Novo_Nodo^.Dado := 13;        { Coloca o dado no nodo que foi alocado         }
    Novo_Nodo^.Pai :=Arvore_Generica;     { O pai do novo nodo é o nodo raiz      }
    Novo_Nodo^.Filho := NIL;      {  Inicialmente o nodo não tem filho            }
    Novo_Nodo^.Irmao := NIL;      {  e também não tem irmão (mais moço)           }
    Ultimo_Filho^.Irmao := Novo_Nodo;     { O irmão aponta p/ novo irmão          }
    Ultimo_Filho := Novo_Nodo;    { Indica que este foi o último filho inserido   }

End.
```

EXERCÍCIOS – *Árvores Genéricas*

1. Faça um programa para a manipulação de árvores genéricas usando listas 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_AG     = ^Nodo_Arv_Gen;
    Nodo_Arv_Gen    = Record
                        Dado: Tipo_Dado;
                        Pai, Filho, Irmao: Ptr_Nodo_AG;
                      End;
```

   Rotinas:

```
{ Inicializa a árvore genérica – Prepara para inserir dados }
Procedure Inicializa_ArvGen (Var Raiz: Ptr_Nodo_AG);

{ Insere um dado na árvore genérica – Insere como filho do nodo indicado }
Procedure Insere_AG_Filho (Var Pai: Ptr_Nodo_AG; Dado: Tipo_Dado);

{ Insere um dado na árvore genérica – Insere como irmao do nodo indicado }
Procedure Insere_AG_Irmao (Var Filho: Ptr_Nodo_AG; Dado: Tipo_Dado);

{ Insere um dado na árvore genérica – Insere como pai do nodo indicado }
Procedure Insere_AG_Pai (Var Filho: Ptr_Nodo_AG; Dado: Tipo_Dado);

{ Exibe o conteúdo (dados) da árvore genérica }
Procedure Exibe_ArvGen (Raiz: Ptr_Nodo_AG);

{ Salva em disco o conteúdo (dados) da árvore genérica }
Procedure Salva_ArvGen (Raiz: Ptr_Nodo_AG);

{ Recupera do disco o conteúdo (dados) da árvore genérica }
Function   Le_ArvGen (Var Raiz: Ptr_Nodo_AG): Boolean;

{ Apaga toda a árvore genérica, liberando a memória ocupada }
Procedure Apaga_ArvGen (Var Raiz: Ptr_Nodo_AG);

{ Procura um dado na árvore e retorna um ponteiro para este dado (ou NIL se não achar) }
Function   Pesquisa_ArvGen (Raiz: Ptr_Nodo_AG; Dado: Tipo_Dado): Ptr_Nodo_AG;

{ Remove o nodo cujo ponteiro foi passado como parâmetro }
Procedure Remove_AG_Nodo (Var Nodo: Ptr_Nodo_AG);

{ Remove o nodo que contem o valor indicado como parâmetro, indicando se conseguiu }
Function   Remove_AG_Dado (Var Raiz: Ptr_Nodo_AG; Dado: Tipo_Dado): Boolean;
```

2. Faça um programa para a manipulação de árvores genéricas usando as rotinas genéricas de manipulação de dados descritas no exercício anterior. Leia um arquivo texto contendo uma seqüência de nomes e coloque os nomes em diferentes níveis da árvore de acordo com a inicial do nome. Exibir na tela a lista de nomes, agrupados de acordo com a letra inicial do nome. Exemplo:

   A
   Andréia
   Ana
   Alberto
   B
   Bianca
   C
   Carla
   Cristina
   ...

3. Faça um programa para a manipulação de árvores genéricas usando as rotinas genéricas de manipulação de dados descritas no exercício anterior. Leia um arquivo texto com uma seqüência de linhas compostas de um identificador e um dado, conforme o exemplo abaixo, e gere uma árvore genérica onde o identificador de cada linha serve para indicar onde esta linha será inserida na hierarquia da árvore. A seguir, gere um arquivo texto, percorrendo a árvore do modo prefixado (ordem a ser seguida: nodo, irmão e por último filho).

   Exemplo do arquivo de dados: (os dígitos de cada sub-item variam apenas de 1 a 9).

   1- Nome
   1.1- Sexo
   1.2- Idade
   1.3- Nacionalidade
   1.3.1- Local_de_Nascimento
   1.3.2- Estado
   1.3.3- CEP
   1.4- Profissão
   1.5- Estado_Civil
   1.6- Carteira_de_Identidade
   1.7- CPF
   1.4.1- Salario
   1.4.2- Cargo