UNISINOS - **UNIVERSIDADE DO VALE DO RIO DOS SINOS**
CIÊNCIAS EXATAS E TECNOLÓGICAS – Curso: Informática / Ciência da Computação

Programação II

Disciplina: Linguagem de Programação PASCAL **E-mail:** ***osorio@exatas.unisinos.br***
Professor responsável: ***Fernando Santos Osório*** **Web:**
Semestre: 2004/2 ***http://www.inf.unisinos.br/~osorio/prog2.html***
Horário: 63 **Xerox : *Pasta 54* – LAB. II (Xerox do "Alemão")**

ALOCAÇÃO DINÂMICA – ÀRVORES BINÀRIAS

Estruturas de Dados com Alocação Dinâmica

As estruturas de dados, que utilizam alocação dinâmica de memória, permitem que os dados sejam organizados de diferentes modos, como por exemplo:

- Listas de dados, que definem uma seqüência de dados;
- Árvores, que definem uma hierarquia de dados (em níveis);
- Grafos, que definem uma rede de dados (rede de interconexões).

Após nós termos trabalhado com as Listas Encadeadas (simplesmente ou duplamente encadeadas), vamos abordar agora as ***árvores*** com alocação dinâmica, uma estrutura que permite organizar os dados em uma forma mais estruturada. Existem diferentes tipos de árvores, onde vamos inicialmente apresentar os conceitos que definem as árvores de um modo geral, focando posteriormente nossa atenção em um tipo particular de árvore: as ***árvores binárias***.

Árvores com Alocação Dinâmica

As árvores são estruturas de dados criadas usualmente através do uso de alocação dinâmica de memória, sendo similares as listas encadeadas, mas no entanto permitindo uma organização hierárquica (em níveis) das informações. As árvores devem possuir um nodo superior, denominado de **nodo raiz**. O nodo raiz também será um **nodo pai**, ou seja, aquele nodo que pode ter associado a ele **nodos filhos**, onde esta associação é feita através de ponteiros que apontam do pai para o(s) filho(s). Por sua vez cada nodo filho pode se tornar um nodo pai e possuir (apontar) para outros nodos filhos, e assim chegamos a uma definição "recursiva" de uma árvore: um nodo pai aponta para nodos filhos, onde estes nodos filhos também podem ser nodos pais, que apontam para nodos filhos, e assim por diante. As "extremidades" de uma árvore, ou seja, os nodos que não possuem mais filhos associados a eles são chamados de **nodos folha**.

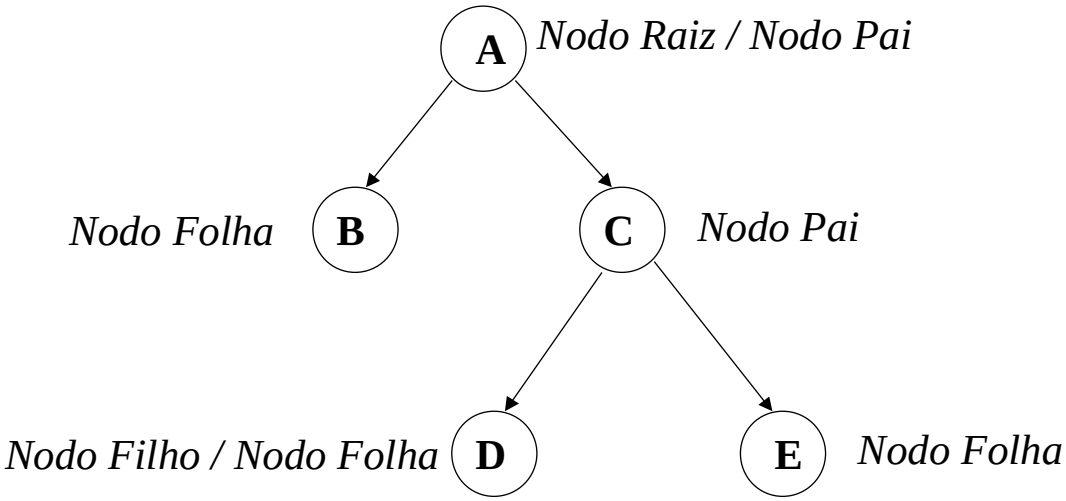
Existem diferentes tipos de árvores, que são classificadas de acordo com a quantidade de filhos por nodo (exemplo: árvore binária é aquela que tem no máximo 2 filhos por nodo) e a disposição dos nodos (exemplo: árvore cujos nodos respeitam algum tipo de ordenação). Os tipos usuais de árvores são:

Árvore ➔ Árvore Binária ➔ Árvore Binária Ordenada (ABO) ➔ Árvore (ABO) Balanceada
Árvore Ternária Árvore Binária Não Ordenada Árvore Binária Não Balanceada
...
Árvore Genérica (pode ter qualquer quantidade de nodos filhos associados a cada nodo)

Árvores Binárias

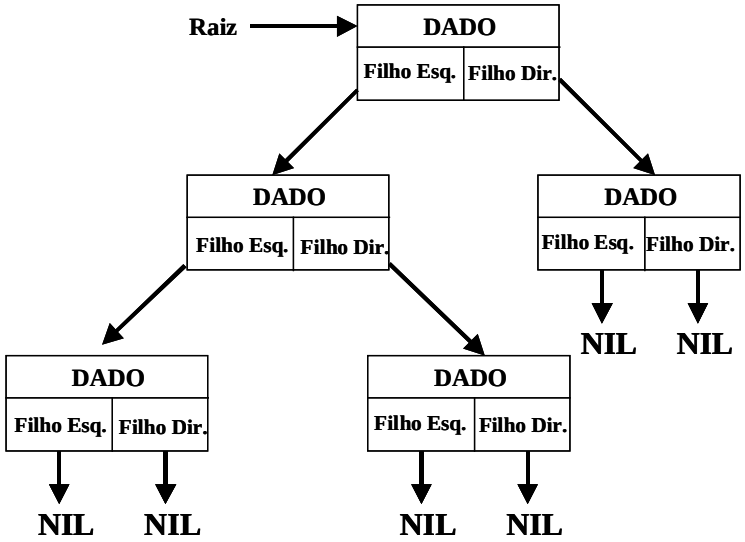
Uma ***árvore binária*** é um tipo específico de árvore que possui apenas 2 nodos filhos ligados a cada nodo pai: o ***nodo da esquerda*** e o ***nodo da direita***. As árvores binárias ainda podem ser do tipo ordenado ou não e balanceada ou não. As árvores binárias ordenadas permitem uma busca bem mais eficiente dos dados, desde que estes tenham sido armazenados de forma ordenada.

A figura abaixo mostra um exemplo de uma árvore binária:

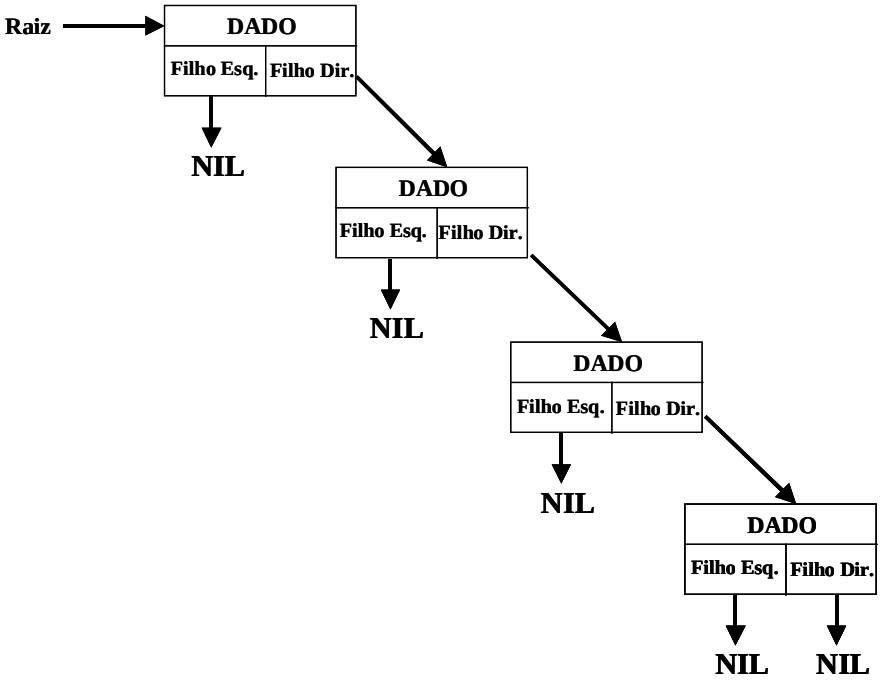


Uma árvore binária possui um nodo contendo dois ponteiros, dividindo os dados em dois grupos. Sendo assim, os nodos de uma árvore binária possuem um ponteiro da esquerda, que aponta para um conjunto de "filhos do lado esquerdo" deste nodo e um ponteiro da direita, que aponta para um conjunto de "filhos do lado direito". Se considerarmos os nodos de uma lista duplamente encadeada e de uma árvore, constatamos que ambos possuem um campo de dado e dois ponteiros, mas a diferença está no fato que o nodo da lista encadeada define uma seqüência (anterior e próximo) e a árvore define uma hierarquia (filho da esquerda e filho da direita, ambos em um nível mais abaixo do nível do pai).

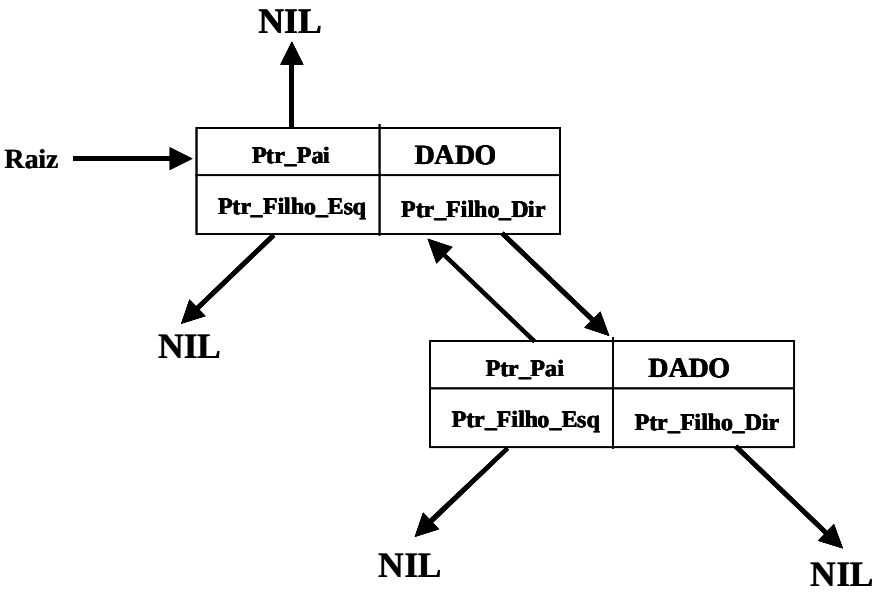
A figura abaixo mostra exemplos de nodos de uma árvore binária:



Note que uma árvore "degenerada" (não balanceada), que possui apenas nodos filhos de um tipo, pode vir a se tornar uma lista encadeada como as estudadas anteriormente:

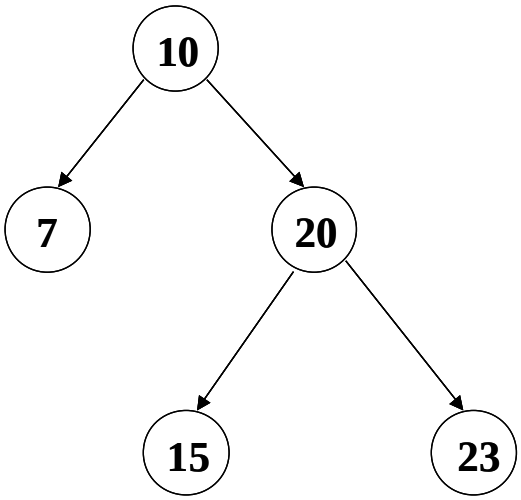


Assim como as listas encadeadas, que podem possui um encadeamento simples ou duplo entre os seus nodos, as árvores também podem possuir um encadeamento seguindo somente em um sentido da hierarquia, do nodo pai para o nodo filho, ou podem também possuir um encadeamento duplo, nos dois sentidos da hierarquia, do nodo pai para o nodo filho e do nodo filho para o nodo pai. No caso do duplo encadeamento, teremos então três ponteiros por nodo, resultando em um nodo com: *dado*, ponteiro para a sub-árvore a esquerda (*filho esq*), ponteiro para a sub-árvore a direita (*filho dir*) e ponteiro para o nodo do nível acima (*pai*), conforme ilustrado na figura abaixo:

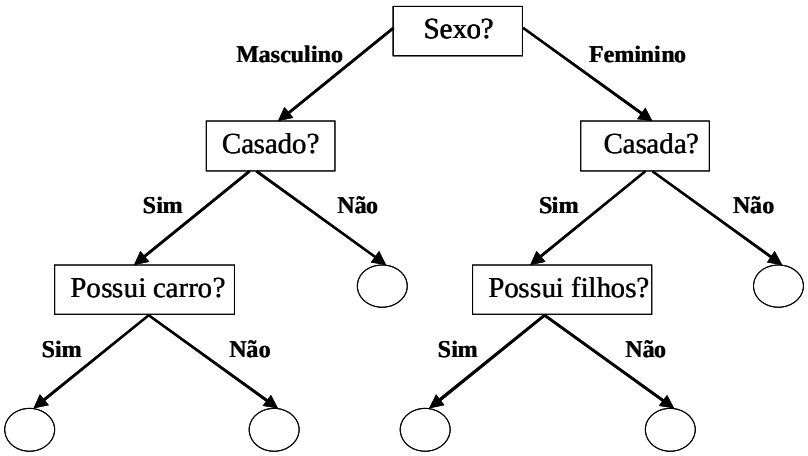


Em relação as funções de manipulação dos dados de uma árvore binária, podemos considerar que existem diferentes tipos de inserção, remoção bem como consulta e exibição do conteúdo de uma árvore. A inserção poderá ou não ser ordenada, poderá ou não gerar uma árvore balanceada (equilibrada e com o menor número de níveis possível), poderá ou não aceitar dados duplicados, etc. A remoção também pode considerar a ordenação e o balanceamento, ou não.

A inserção ordenada de dados em uma árvore binária adota a seguinte regra: os dados menores (ou maiores) que um determinado nodo devem ser colocados em sua sub-árvore esquerda, e os dados maiores (ou menores) que um determinado nodo devem ser colocados em sua sub-árvore direita. Se for garantida esta regra, menores de um lado e maiores do outro, para todos os nodos da árvore, teremos então uma árvore binária ordenada. A ordenação será usualmente garantida através de uma inserção ordenada, que localiza e insere o nodo na sua posição correta. A figura abaixo apresenta um exemplo de árvore binária ordenada (a) e de árvore binária desordenada (b).



(a) Árvore Binária Ordenada



(b) Árvore Binária Desordenada

Além das operações de inserção e remoção, uma outra função muito importante no que se refere as árvores binárias, são as operações de busca e exibição do conteúdo da árvore. Uma árvore binária (ordenada ou não) poderá ser percorrida das seguintes maneiras:

- Percorrer de **modo préfixado**: Pré-Ordem (VED = Visita/Esquerda/Direita);
- Percorrer de **modo infixado** : Em Ordem (EVD = Esquerda/Visita/Direita);
- Percorrer de **modo pósfixado**: Pós-Ordem (EDV = Esquerda/Direita/Visita);
-
- Em **modo de Busca Completa** (percorre todo os nodos)
- Em **modo de Busca Binária** (*apenas em árvores binárias ordenadas!*)

Vamos definir a seguir um conjunto de **rotinas genéricas** para manipulação de **Árvores Binárias Ordenadas (ABO)**, que simplificarão o seu uso e sua adaptação para outras aplicações. Foi definido nesta implementação a adoção de um nodo com ponteiro para o pai (duplo encadeamento) e inicialmente não iremos tratar do balanceamento da árvore (ABO não balanceada).

Definição dos Dados:

```
Type
    TArvBin_Dado  =  Integer;

    Ptr_ArvBin_Nodo  =  ^ArvBin_Nodo;

    ArvBin_Nodo       =  Record
                            Dado: TArvBin_Dado;
                            Pai: Ptr_ArvBin_Nodo;
                            ArvEsq, ArvDir : Ptr_ArvBin_Nodo;
                         End;
```

Definição das Rotinas:

```
Procedure ArvBin_Inicializa (Var ABO: Ptr_ArvBin_Nodo);
{   ABO  = Ponteiro para a raiz da árvore binária ordenada, inicializado com NIL  }
Begin
    ABO := NIL;
End;
```

```
Procedure ArvBin_Insere_Ordenado (Var ABO: Ptr_ArvBin_Nodo; Dado: TArvBin_Dado);
{   Insere um dado de modo ordenado  na árvore binária    }
Var
   novo, ptr, ant : Ptr_ArvBin_Nodo;
Begin
   New(novo);
   novo^.dado := Dado;
   novo^.ArvEsq := NIL;
   novo^.ArvDir := NIL;
   ptr := ABO;
```

```
  While  ptr <> NIL
  Do  Begin
          ant := ptr;
          If  dado > ptr^.dado
          Then ptr := ptr^.ArvDir
          Else  ptr := ptr^.ArvEsq;
      End;
   If  ptr = ABO
   Then Begin
           novo^.Pai := NIL;
           ABO := novo;
        End
   Else   Begin
            novo^.Pai := ant;
            If  dado > ant^.dado
            Then ant^.ArvDir := novo
            Else   ant^.ArvEsq := novo;
        End;
End;
```

Procedure ArvBin_Exibe_Infixado (ABO: Ptr_ArvBin_Nodo);

```
{ Exibe o conteúdo (dados) da árvore em ordem infixada... seja ela ordenada ou não  }
Begin
   If   ABO <> NIL
   Then  Begin
            ArvBin_Exibe_Infixado(ABO^.ArvEsq);
            Writeln (ABO^.dado);
            ArvBin_Exibe_Infixado(ABO^.ArvDir);
         End;
End;
```

Procedure ArvBin_Exibe_Prefixado (ABO: Ptr_ArvBin_Nodo);

```
{ Exibe o conteúdo (dados) da árvore em ordem prefixada... seja ela ordenada ou não  }
Begin
   If   ABO <> NIL
   Then  Begin
            Writeln (ABO^.dado);
            ArvBin_Exibe_Prefixado(ABO^.ArvEsq);
            ArvBin_Exibe_Prefixado(ABO^.ArvDir);
         End;
End;
```

Procedure ArvBin_Exibe_Posfixado (ABO: Ptr_ArvBin_Nodo);

```
{ Exibe o conteúdo (dados) da árvore em ordem posfixada... seja ela ordenada ou não  }
Begin
   If   ABO <> NIL
   Then  Begin
            ArvBin_Exibe_Posfixado(ABO^.ArvEsq);
            ArvBin_Exibe_Posfixado(ABO^.ArvDir);
            Writeln (ABO^.dado);
         End;
End;
```