5.3.1 Árvores Binárias de Pesquisa sem Balanceamento

De acordo com Knuth (1997, p. 312), uma **árvore binária** é definida como um conjunto finito de nós que ou está vazio ou consiste de um nó chamado raiz mais os elementos de duas árvores binárias distintas chamadas de subárvores esquerda e direita do nó raiz. Em uma árvore binária, cada nó tem no máximo duas subárvores.

Existem apontadores para as subárvores esquerda e direita em cada nó. O número de subárvores de um nó é chamado grau daquele nó. Um nó de grau zero é chamado de nó externo ou folha (de agora em diante não haverá distinção entre esses dois termos). Os outros nós são chamados nós internos.

A **árvore binária de pesquisa** é uma árvore binária em que todo nó interno contém um registro, e, para cada nó, a seguinte propriedade é verdadeira: todos os registros com chaves menores estão na subárvore esquerda e todos os registros com chaves maiores estão na subárvore direita.

O **nível** do nó raiz é 0; se um nó está no nível i então a raiz de suas subárvores está no nível $i + 1$. A **altura** de um nó é o comprimento do caminho mais longo deste nó até um nó folha. A altura de uma árvore é a altura do nó raiz. A Figura 5.2 mostra uma árvore binária de pesquisa de altura 4.

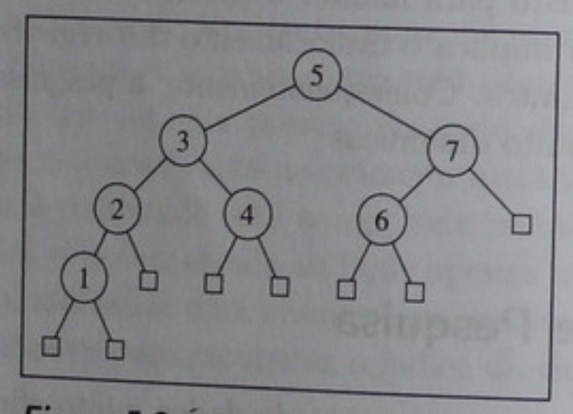

Figura 5.2 *Árvore binária de pesquisa.*

A estrutura de dados árvore binária de pesquisa será utilizada para implementar o tipo abstrato de dados Dicionário (lembre-se que o tipo abstrato Dicionário contém as operações Inicializa, Pesquisa, Insere e Retira). A estrutura e a **representação** do Dicionário são apresentadas no Programa 5.4.

Um procedimento Pesquisa para uma árvore binária de pesquisa é bastante simples, conforme ilustra a implementação do Programa 5.5. Para encontrar um registro que contém a chave x, primeiro compare-a com a chave que está na raiz. Se é menor, vá para a subárvore esquerda; se x é maior, vá para a subárvore direita. Repita o processo recursivamente, até que a chave procurada seja encontrada ou então um nó folha é atingido. Se a pesquisa tiver sucesso, então o conteúdo do registro retorna no próprio registro x.

Programa 5.4 *Estrutura do dicionário para árvores sem balanceamento*

```
type TipoChave      = integer;
     TipoRegistro   = record
                        Chave: TipoChave;
                        { outros componentes }
                      end;
     TipoApontador  = ^TipoNo;
     TipoNo         = record
                        Reg: TipoRegistro;
                        Esq, Dir: TipoApontador;
                      end;
     TipoDicionario = TipoApontador;
```

Programa 5.5 *Procedimento para pesquisar na árvore*

```
procedure Pesquisa (var x: TipoRegistro; var p: TipoApontador);
begin
  if p = nil
  then writeln ('Erro: TipoRegistro nao esta presente na arvore')
  else if x.Chave < p^.Reg.Chave
       then Pesquisa (x, p^.Esq)
       else if x.Chave > p^.Reg.Chave
            then Pesquisa (x, p^.Dir)
            else x := p^.Reg;
end; { Pesquisa }
```

O procedimento Inicializa é extremamente simples, conforme ilustra o Programa 5.6.

Programa 5.6 *Procedimento para inicializar*

```
procedure Inicializa (var Dicionario: TipoDicionario);
begin
  Dicionario := nil;
end; { Inicializa }
```

Atingir um apontador nulo em um processo de pesquisa significa uma pesquisa sem sucesso (o registro procurado não está na árvore). Caso se queira inseri-lo na árvore, o apontador nulo atingido é justamente o ponto de inserção, conforme ilustra a implementação do procedimento **Insere** do Programa 5.7.