

**Programa 5.16** Estrutura de dados

```

const D = 8; { depende de TipoChave }
type TipoChave = char; { a definir, dependendo da aplicacao }
TipoIndexAmp = 0..D;
TipoDib      = 0..1;
TipoNo       = (Interno, Externo);
TipoArvore   = ^TipoPatNo;
TipoPatNo    = record
    case nt: TipoNo of
        Interno: (Index: TipoIndexAmp; Esq, Dir: TipoArvore);
        Externo: (Chave: TipoChave);
    end;
end;

```

**Programa 5.17** Funções auxiliares

```

function Bit (i: TipoIndexAmp; k: TipoChave): TipoDib;
{ Retorna o i-ésimo bit da chave k a partir da esquerda }
var c, j: integer;
begin
    if i = 0
    then Bit := 0
    else begin
        c := ord (k);
        for j := 1 to D - i do c := c div 2;
        Bit := c mod 2;
        end;
    end; { Bit }
end;

```

```

function EExterno (p: TipoArvore): boolean;
{ Verifica se p é nodo externo }
begin
    EExterno := p^.nt = Externo;
end; { EExterno }

```

**Programa 5.18** Procedimento para criar nó interno

```

function CriaNodoInt (i: integer; var Esq, Dir: TipoArvore): TipoArvore;
var p: TipoArvore;
begin
    new (p, Interno);
    p^.nt := Interno;
    p^.Esq := Esq; p^.Dir := Dir;
    p^.Index := i; CriaNodoInt := p;
end; { CriaNodoInt }

```

**Programa 5.19** Procedimento para criar nó externo

```

function CriaNodoExt (k: TipoChave): TipoArvore;
var p: TipoArvore;
begin
    new (p, Externo);
    p^.nt := Externo;
    p^.Chave := k;
    CriaNodoExt := p;
end; { CriaNodoExt }

```

**Programa 5.20** Algoritmo de pesquisa

```

procedure Pesquisa (k: TipoChave; t: TipoArvore);
begin
    if EExterno (t)
    then if k = t^.Chave
        then writeln ('Elemento encontrado')
        else writeln ('Elemento nao encontrado')
    else if Bit (t^.Index, k) = 0
        then Pesquisa (k, t^.Esq)
        else Pesquisa (k, t^.Dir)
    end; { Pesquisa }
end;

```

**Programa 5.21** Algoritmo de inserção

```

function Insere (k: TipoChave; var t: TipoArvore): TipoArvore;
var p: TipoArvore; i: integer;
function InsereEntre (k: TipoChave; var t: TipoArvore;
    i: integer): TipoArvore;
var p: TipoArvore;
begin
    if EExterno (t) or (i < t^.Index)
    then begin { cria um novo no externo }
        p := CriaNodoExt (k);
        if Bit (i, k) = 1
        then InsereEntre := CriaNodoInt (i, t, p)
        else InsereEntre := CriaNodoInt (i, p, t);
        end
    else begin
        if Bit (t^.Index, k) = 1
        then t^.Dir := InsereEntre (k, t^.Dir, i)
        else t^.Esq := InsereEntre (k, t^.Esq, i);
        InsereEntre := t;
        end;
    end; { InsereEntre }
end;

```