

## Estrutura de Dados

```
#define D 8 /* depende de TipoChave */
typedef unsigned char TipoChave; /* a definir */
typedef unsigned char TipoIndexAmp;
typedef unsigned char TipoDib;
typedef enum {
    Interno, Externo
} TipoNo;
typedef struct TipoPatNo* TipoArvore;
typedef struct TipoPatNo {
    TipoNo nt;
    union {
        struct {
            TipoIndexAmp Index;
            TipoArvore Esq, Dir;
        } NInterno;
        TipoChave Chave;
    } NO;
} TipoPatNo;
```

## Funções Auxiliares

```
TipoDib Bit(TipoIndexAmp i, TipoChave k)
{ /* Retorna o i-esimo bit da chave k a partir da esquerda */
    int c, j;
    if (i == 0)
        return 0;
    else { c = k;
        for (j = 1; j <= D - i; j++) c /= 2;
        return (c & 1);
    }
}

short EExterno(TipoArvore p)
{ /* Verifica se p^ e nodo externo */
    return (p->nt == Externo);
}
```

Fonte:

(ZIVIANI, 2010)

## Procedimentos para Criar Nós Interno e Externo

```
TipoArvore CriaNoInt(int i, TipoArvore *Esq, TipoArvore *Dir)
{ TipoArvore p;
  p = (TipoArvore)malloc(sizeof(TipoPatNo));
  p->nt = Interno; p->NO.NInterno.Esq = *Esq;
  p->NO.NInterno.Dir = *Dir; p->NO.NInterno.Index = i;
  return p;
}
```

```
TipoArvore CriaNoExt(TipoChave k)
{ TipoArvore p;
  p = (TipoArvore)malloc(sizeof(TipoPatNo));
  p->nt = Externo; p->NO.Chave = k; return p;
}
```

## Algoritmo de Pesquisa

```
void Pesquisa(TipoChave k, TipoArvore t)
{ if (EExterno(t))
  { if (k == t->NO.Chave)
    printf("Elemento encontrado\n");
    else printf("Elemento nao encontrado\n");
    return;
  }
  if (Bit(t->NO.NInterno.Index, k) == 0)
    Pesquisa(k, t->NO.NInterno.Esq);
  else Pesquisa(k, t->NO.NInterno.Dir);
}
```

## Algoritmo de inserção

```
TipoArvore InsereEntre(TipoChave k, TipoArvore *t, int i)
{ TipoArvore p;
  if (EExterno(*t) || i < (*t)->NO.NInterno.Index)
  { /* cria um novo no externo */
    p = CriaNoExt(k);
    if (Bit(i, k) == 1)
        return (CriaNoInt(i, t, &p));
    else return (CriaNoInt(i, &p, t));
  }
  else
  { if (Bit((*t)->NO.NInterno.Index, k) == 1)
    (*t)->NO.NInterno.Dir = InsereEntre(k, &(*t)->NO.NInterno.Dir, i);
    else
    (*t)->NO.NInterno.Esq = InsereEntre(k, &(*t)->NO.NInterno.Esq, i);
    return (*t);
  }
}
```

## Algoritmo de inserção

```
TipoArvore Insere(TipoChave k, TipoArvore *t)
{ TipoArvore p; int i;
  if (*t == NULL) return (CriaNoExt(k));
  else
  { p = *t;
    while (!EExterno(p))
    { if (Bit(p->NO.NInterno.Index, k) == 1) p = p->NO.NInterno.Dir;
      else p = p->NO.NInterno.Esq;
    }
    /* acha o primeiro bit diferente */
    i = 1;
    while ((i <= D) & (Bit((int)i, k) == Bit((int)i, p->NO.Chave)))
        i++;
    if (i > D) { printf("Erro: chave ja esta na arvore\n"); return (*t); }
    else return (InsereEntre(k, t, i));
  }
}
```