Programa E.16 *Estrutura de dados*

```
#define D 8   /* depende de TipoChave */

typedef unsigned char TipoChave; /* a definir, depende da aplicacao */
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;
```

Programa E.17 *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);
}
```

Programa E.18 *Procedimento para criar nó interno*

```
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;
}
```


Programa E.19 *Procedimento para criar nó externo*

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

Programa E.20 *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);
}
```

Programa E.21 *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);
  }
}

TipoArvore Insere(TipoChave k, TipoArvore *t)
{ TipoArvore p;
  int i;
  if (*t == NULL)
  return (CriaNoExt(k));
  else
  { p = *t;
```