Continuação do Programa F.9

```
void Antecessor(TipoApontador Ap, int Ind,
                TipoApontador ApPai, short *Diminuiu)
{ if (ApPai->p[ApPai->n] != NULL)
  { Antecessor(Ap, Ind, ApPai->p[ApPai->n], Diminuiu);
    if (*Diminuiu)
    Reconstitui(ApPai->p[ApPai->n], ApPai, (long)ApPai->n, Diminuiu);
    return;
  }
  Ap->r[Ind-1] = ApPai->r[ApPai->n - 1];
  ApPai->n--; *Diminuiu = (ApPai->n < M);
}

void Ret(TipoChave Ch, TipoApontador *Ap, short *Diminuiu)
{ long j, Ind = 1;
  TipoApontador Pag;
  if (*Ap == NULL)
  { printf("Erro: registro nao esta na arvore\n"); *Diminuiu = FALSE;
    return;
  }
  Pag = *Ap;
  while (Ind < Pag->n && Ch > Pag->r[Ind-1].Chave) Ind++;
  if (Ch == Pag->r[Ind-1].Chave)
  { if (Pag->p[Ind-1] == NULL)    /* TipoPagina folha */
    { Pag->n--;
      *Diminuiu = (Pag->n < M);
      for (j = Ind; j <= Pag->n; j++)
        { Pag->r[j-1] = Pag->r[j]; Pag->p[j] = Pag->p[j+1]; }
      return;
    }
    /* TipoPagina nao e folha: trocar com antecessor */
    Antecessor(*Ap, Ind, Pag->p[Ind-1], Diminuiu);
    if (*Diminuiu)
    Reconstitui(Pag->p[Ind-1], *Ap, Ind - 1, Diminuiu);
    return;
  }
  if (Ch > Pag->r[Ind-1].Chave) Ind++;
  Ret(Ch, &Pag->p[Ind-1], Diminuiu);
  if (*Diminuiu) Reconstitui(Pag->p[Ind-1], *Ap, Ind - 1, Diminuiu);
}

void Retira(TipoChave Ch, TipoApontador *Ap)
{ short Diminuiu;
  TipoApontador Aux;
  Ret(Ch, Ap, &Diminuiu);
  if (Diminuiu && (*Ap)->n == 0)  /* Arvore diminui na altura */
  { Aux = *Ap;    *Ap = Aux->p[0];
    free(Aux);
  }
}
```


Programa F.10 *Estrutura do dicionário para a árvore B**

```
typedef int TipoChave;
typedef struct TipoRegistro {
  TipoChave Chave;
  /* outros componentes */
} TipoRegistro;
typedef enum {
  Interna, Externa
} TipoIntExt;
typedef struct TipoPagina *TipoApontador;
typedef struct TipoPagina {
  TipoIntExt Pt;
  union {
    struct {
      int ni;
      TipoChave ri[MM];
      TipoApontador pi[MM + 1];
    } U0;
    struct {
      int ne;
      TipoRegistro re[MM2];
    } U1;
  } UU;
} TipoPagina;
```

Programa F.11 *Função para pesquisar na árvore B**

```
void Pesquisa(TipoRegistro *x, TipoApontador *Ap)
{ int i;
  TipoApontador Pag;
  Pag = *Ap;
  if ((*Ap)->Pt == Interna)
  { i = 1;
    while (i < Pag->UU.U0.ni && x->Chave > Pag->UU.U0.ri[i - 1]) i++;
    if (x->Chave < Pag->UU.U0.ri[i - 1])
    Pesquisa(x, &Pag->UU.U0.pi[i - 1]);
    else Pesquisa(x, &Pag->UU.U0.pi[i]);
    return;
  }
  i = 1;
  while (i < Pag->UU.U1.ne && x->Chave > Pag->UU.U1.re[i - 1].Chave)
    i++;
  if (x->Chave == Pag->UU.U1.re[i - 1].Chave)
  *x = Pag->UU.U1.re[i - 1];
  else printf("TipoRegistro nao esta presente na arvore\n");
}
```