

Continuação do Programa E.13

```

Insere(x, &(*Ap)->Dir, &(*Ap)->BitD, Fim);
if (*Fim) return;
if ((*Ap)->BitD != Horizontal) { *Fim = TRUE; return; }
if ((*Ap)->Dir->BitD == Horizontal)
{ DD(Ap); *Lap = Horizontal; return; }
if ((*Ap)->Dir->BitE == Horizontal) { DE(Ap); *Lap = Horizontal; }
}

void Insere(TipoRegistro x, TipoApontador *Ap)
{ short Fim; TipoInclinacao Lap;
  Insere(x, Ap, &Lap, &Fim);
}

```

Programa E.14 Procedimento para inicializar a árvore SBB

```

void Inicializa(TipoApontador *Dicionario)
{ *Dicionario = NULL; }

```

Programa E.15 Procedimento para retirar da árvore SBB

```

void EsqCurto(TipoApontador *Ap, short *Fim)
{ /* Folha esquerda retirada => arvore curta na altura esquerda */
  TipoApontador Apl;
  if ((*Ap)->BitE == Horizontal)
  { (*Ap)->BitE = Vertical; *Fim = TRUE; return; }
  if ((*Ap)->BitD == Horizontal)
  { Apl = (*Ap)->Dir; (*Ap)->Dir = Apl->Esq; Apl->Esq = *Ap; *Ap = Apl;
    if ((*Ap)->Esq->Dir->BitE == Horizontal)
    { DE(&(*Ap)->Esq); (*Ap)->BitE = Horizontal; }
    else if ((*Ap)->Esq->Dir->BitD == Horizontal)
    { DD(&(*Ap)->Esq); (*Ap)->BitE = Horizontal; }
    *Fim = TRUE; return;
  }
  (*Ap)->BitD = Horizontal;
  if ((*Ap)->Dir->BitE == Horizontal) { DE(Ap); *Fim = TRUE; return; }
  if ((*Ap)->Dir->BitD == Horizontal) { DD(Ap); *Fim = TRUE; }
}

void DirCurto(TipoApontador *Ap, short *Fim)
{ /* Folha direita retirada => arvore curta na altura direita */
  TipoApontador Apl;
  if ((*Ap)->BitD == Horizontal)
  { (*Ap)->BitD = Vertical; *Fim = TRUE; return; }
  if ((*Ap)->BitE == Horizontal)
  { Apl = (*Ap)->Esq; (*Ap)->Esq = Apl->Dir; Apl->Dir = *Ap; *Ap = Apl;
    if ((*Ap)->Dir->Esq->BitD == Horizontal)
    { ED(&(*Ap)->Dir); (*Ap)->BitD = Horizontal; }
  }
}

```

Continuação do Programa E.15

```

else if ((*Ap)->Dir->Esq->BitE == Horizontal)
{ EE(&(*Ap)->Dir); (*Ap)->BitD = Horizontal; }
*Fim = TRUE; return;
}
(*Ap)->BitE = Horizontal;
if ((*Ap)->Esq->BitD == Horizontal) { ED(Ap); *Fim = TRUE; return; }
if ((*Ap)->Esq->BitE == Horizontal) { EE(Ap); *Fim = TRUE; }
}

void Antecessor(TipoApontador q, TipoApontador *r, short *Fim)
{ if ((*r)->Dir != NULL)
{ Antecessor(q, &(*r)->Dir, Fim);
  if (!*Fim) DirCurto(r, Fim); return;
}
q->Reg = (*r)->Reg; q = *r; *r = (*r)->Esq; free(q);
if (*r != NULL) *Fim = TRUE;
}

void IRetira(TipoRegistro x, TipoApontador *Ap, short *Fim)
{ TipoNo *Aux;
  if (*Ap == NULL)
  { printf("Chave nao esta na arvore\n"); *Fim = TRUE; return; }
  if (x.Chave < (*Ap)->Reg.Chave)
  { IRetira(x, &(*Ap)->Esq, Fim);
    if (!*Fim) EsqCurto(Ap, Fim); return;
  }
  if (x.Chave > (*Ap)->Reg.Chave)
  { IRetira(x, &(*Ap)->Dir, Fim);
    if (!*Fim) DirCurto(Ap, Fim); return;
  }
  *Fim = FALSE; Aux = *Ap;
  if (Aux->Dir == NULL)
  { *Ap = Aux->Esq; free(Aux);
    if (*Ap != NULL) *Fim = TRUE; return;
  }
  if (Aux->Esq == NULL)
  { *Ap = Aux->Dir; free(Aux);
    if (*Ap != NULL) *Fim = TRUE; return;
  }
  Antecessor(Aux, &Aux->Esq, Fim);
  if (!*Fim) EsqCurto(Ap, Fim); /* Encontrou chave */
}

void Retira(TipoRegistro x, TipoApontador *Ap)
{ short Fim;
  IRetira(x, Ap, &Fim);
}

```