*Continuação do Programa E.9*

```
void Retira(TipoRegistro x, TipoApontador *p)
{  TipoApontador Aux;
   if (*p == NULL)
   { printf("Erro : Registro nao esta na arvore\n");
     return;
   }
   if (x.Chave < (*p)->Reg.Chave) { Retira(x, &(*p)->Esq); return; }
   if (x.Chave > (*p)->Reg.Chave) { Retira(x, &(*p)->Dir); return; }
   if ((*p)->Dir == NULL)
   { Aux = *p;   *p = (*p)->Esq;
     free(Aux);
     return;
   }
   if ((*p)->Esq != NULL)
   { Antecessor(*p, &(*p)->Esq);
     return;
   }
   Aux = *p;   *p = (*p)->Dir;
   free(Aux);
}
```

***Programa E.10*** *Caminhamento central*

```
void Central(TipoApontador p)
{ if (p == NULL) return;
   Central(p->Esq);
   printf("%ld\n", p->Reg.Chave);
   Central(p->Dir);
}
```

***Programa E.11*** *Estrutura do dicionário para árvores SBB*

```
typedef int TipoChave;
typedef struct TipoRegistro {
  /* outros componentes */
  TipoChave Chave;
} TipoRegistro;
typedef enum {
  Vertical, Horizontal
} TipoInclinacao;
typedef struct TipoNo* TipoApontador;
typedef struct TipoNo {
  TipoRegistro Reg;
  TipoApontador Esq, Dir;
  TipoInclinacao BitE, BitD;
} TipoNo;
```



***Programa E.12*** *Procedimentos auxiliares para árvores SBB*

```
void EE(TipoApontador *Ap)
{ TipoApontador Ap1;
  Ap1 = (*Ap)->Esq; (*Ap)->Esq = Ap1->Dir; Ap1->Dir = *Ap;
  Ap1->BitE = Vertical; (*Ap)->BitE = Vertical; *Ap = Ap1;
}
void ED(TipoApontador *Ap)
{ TipoApontador Ap1, Ap2;
  Ap1 = (*Ap)->Esq; Ap2 = Ap1->Dir; Ap1->BitD = Vertical;
  (*Ap)->BitE = Vertical; Ap1->Dir = Ap2->Esq; Ap2->Esq = Ap1;
  (*Ap)->Esq = Ap2->Dir; Ap2->Dir = *Ap; *Ap = Ap2;
}
void DD(TipoApontador *Ap)
{ TipoApontador Ap1;
  Ap1 = (*Ap)->Dir; (*Ap)->Dir = Ap1->Esq; Ap1->Esq = *Ap;
  Ap1->BitD = Vertical; (*Ap)->BitD = Vertical; *Ap = Ap1;
}
void DE(TipoApontador *Ap)
{ TipoApontador Ap1, Ap2;
  Ap1 = (*Ap)->Dir; Ap2 = Ap1->Esq; Ap1->BitE = Vertical;
  (*Ap)->BitD = Vertical; Ap1->Esq = Ap2->Dir; Ap2->Dir = Ap1;
  (*Ap)->Dir = Ap2->Esq; Ap2->Esq = *Ap; *Ap = Ap2;
}
```

***Programa E.13*** *Procedimento para inserir na árvore SBB*

```
void IInsere(TipoRegistro x, TipoApontador *Ap,
             TipoInclinacao *IAp, short *Fim)
{ if (*Ap == NULL)
  { *Ap = (TipoApontador)malloc(sizeof(TipoNo));
    *IAp = Horizontal;   (*Ap)->Reg = x;
    (*Ap)->BitE = Vertical;   (*Ap)->BitD = Vertical;
    (*Ap)->Esq = NULL; (*Ap)->Dir = NULL; *Fim = FALSE;
    return;
  }
  if (x.Chave < (*Ap)->Reg.Chave)
  { IInsere(x, &(*Ap)->Esq, &(*Ap)->BitE, Fim);
    if (*Fim) return;
    if ((*Ap)->BitE != Horizontal) { *Fim = TRUE; return; }
    if ((*Ap)->Esq->BitE == Horizontal)
    { EE(Ap); *IAp = Horizontal; return; }
    if ((*Ap)->Esq->BitD == Horizontal) { ED(Ap); *IAp = Horizontal; }
    return;
  }
  if (x.Chave <= (*Ap)->Reg.Chave)
  { printf("Erro: Chave ja esta na arvore\n");
    *Fim = TRUE;
    return;
  }
```