Implementação das Rotações

```
typedef struct _no NO_AVL;
typedef struct _avl AVL;

struct _no{
  float elem;
  NO_AVL *pai;     /* ponteiro para pai */
  NO_AVL *fesq;    /* ponteiro filho da esquerda */
  NO_AVL *fdir;    /* ponteiro filho da direita */
  int bal;         /* fator de balanceamento */
};

struct _abb{
  NO_AVL *raiz;
  int altura;
  int n_elem;
};

/* rotação tipo LL */
NO_AVL *rotacao_LL(NO_AVL *desb)
{
  NO_AVL *aux;

  aux = desb->fesq;
  if (desb->pai)            /* verifica se desb não é a raiz */
         if (desb->pai->fesq == desb)
                 desb->pai->fesq = aux;
         else
                desb->pai->fdir = aux;

  aux->pai = desb->pai;
  desb->fesq = aux->fdir;
  if (desb->fesq)
         desb->fesq->pai = desb;
  aux->fdir = desb;
  desb->pai = aux;

  return(aux);
}

/* rotação tipo RR */
NO_AVL *rotacao_RR(NO_AVL *desb)
{
  NO_AVL *aux;

  aux = desb->fdir;
  if (desb->pai)            /* verifica se desb não é a raiz */
         if (desb->pai->fesq == desb)
                 desb->pai->fesq = aux;
         else
                desb->pai->fdir = aux;

  aux->pai = desb->pai;
  desb->fdir = aux->fesq;
  if (desb->fdir)
         desb->fdir->pai = desb;
  aux->fesq = desb;
  desb->pai = aux;

  return(aux);
}

/* rotação tipo LR */
NO_AVL *rotacao_LR(NO_AVL *desb)
{
        rotacao_RR(desb->fesq);
        return(rotacao_LL(desb));
}

/* rotação tipo RL */
NO_AVL *rotacao_RL(NO_AVL *desb)
{
        rotacao_LL(desb->fdir);
        return(rotacao_RR(desb));
}
```

« Inserção em uma Árvore AVL