```
tipo
  ref  =  ↑nodo
  nodo  =  registro
           chave             : inteiro
           subArvoreEsquerda,
           subArvoreDireita    : ref
         fim registro

procedimento insere (elemento : inteiro; var p : ref)
início
  se p = nulo então
    aloque (p)
    p↑.chave ← elemento
    p↑.subArvoreEsquerda ← nulo
    p↑.subArvoreDireita ← nulo
  senão
    se elemento < p↑.chave então
      insere (elemento, p↑.subArvoreEsquerda)
    senão
      insere (elemento, p↑.subArvoreDireita)
    fim se
  fim se
fim
```

```
algoritmo arvoreBalanceada

tipo
  ref  =  ↑nodo
  nodo  =  registro
           chave             : inteiro
           subArvoreEsquerda,
           subArvoreDireita    : ref
         fim registro

variável
  n    : inteiro
  raiz : ref

função constroiArvoreBalanceada (n : inteiro) : ref
variável
  p                             : ref
  elemento, numEsquerda, numDireita   : inteiro
início
  se n = 0 então
    constroiArvoreBalanceada ← nulo
  senão
    numEsquerda ← n div 2
    numDireita ← n - numEsquerda - 1
    leia (elemento)
    aloque (p)
    p↑.chave ← elemento
    p↑.subArvoreEsquerda ← constroiArvoreBalanceada (numEsquerda)
    p↑.subArvoreDireita ← constroiArvoreBalanceada (numDireita)
    constroiArvoreBalanceada ← p
  fim se
fim


início
  leia (n)
  raiz ← constroiArvoreBalanceada (n)
  …
fim
```

```
tipo
  ref  =  ↑nodo
  nodo  =  registro
             chave             : inteiro
             subArvoreEsquerda,
             subArvoreDireita    : ref
           fim registro


procedimento retira (elemento : inteiro; var p : ref)
variável
  q : ref

procedimento retiraElemento (var r : ref)
início
  se r↑.subArvoreDireita ≠ nulo então
    retiraElemento (r↑.subArvoreDireita)
  senão
    q↑.chave ← r↑.chave
    q ← r
    r ← r↑.subArvoreEsquerda
  fim se
fim

início
  se elemento < p↑.chave então
    retira (elemento, p↑.subArvoreEsquerda)
  senão
    se elemento > p↑.chave então
      retira (elemento, p↑.subArvoreDireita)
    senão
      q ← p
      se q↑.subArvoreDireita = nulo então
        p ← q↑.subArvoreEsquerda
      senão
        se q↑.subArvoreEsquerda = nulo então
          p ← q↑.subArvoreDireita
        senão
          retiraElemento (q↑.subArvoreEsquerda)
        fim se
      fim se
      libere (q)
    fim se
  fim se
fim
```