```c
/* Avl.h */
#ifndef _AVL_H
#define _AVL_H

//0--代表比较相同，>0--代表dataAddr<keyAddr，<0--代表dataAddr>keyAddr
typedef int  AvlCmp(const void *keyAddr, const void *dataAddr);
typedef void AvlFree(void *);
typedef void AvlTraverseOp(void *);

typedef struct avl_node
{
    struct avl_node *parent;
    struct avl_node *lc;
    struct avl_node *rc;
    int height;
    char key[0];
}AVLNODE;

typedef struct
{
    AVLNODE *root;
    AVLNODE *hot;//“命中”节点的父亲
    int size;
    int keySize;
    AvlCmp *cmpFn;
    AvlFree *freeFn;
}AVLTREE;

//Avl初始化
void AvlNew(AVLTREE *avlTree, int keySize, AvlCmp *cmpFn, AvlFree *freeFn);
//Avl销毁
void AvlDispose(AVLTREE *avlTree);
//Avl判空
int AvlEmpty(AVLTREE *avlTree);
//Avl规模
int AvlSize(AVLTREE *avlTree);
//Avl树高度
int AvlHeight(AVLTREE *avlTree);
//Avl先序遍历（非递归）
void AvlTravPre(AVLTREE *avlTree, AvlTraverseOp *traverseOpFn);
//Avl先序遍历（递归）
void AvlTravPreRec(AVLTREE *avlTree, AvlTraverseOp *traverseOpFn);
//Avl中序遍历（非递归）
void AvlTravIn(AVLTREE *avlTree, AvlTraverseOp *traverseOpFn);
//Avl中序遍历（递归）
void AvlTravInRec(AVLTREE *avlTree, AvlTraverseOp *traverseOpFn);
//Avl后序遍历（非递归）
void AvlTravPost(AVLTREE *avlTree, AvlTraverseOp *traverseOpFn);
//Avl后序遍历（递归）
void AvlTravPostRec(AVLTREE *avlTree, AvlTraverseOp *traverseOpFn);
//Avl层序遍历
void AvlTravLevel(AVLTREE *avlTree, AvlTraverseOp *traverseOpFn);
//Avl中查找关键码所在节点
AVLNODE *AvlSearch(AVLTREE *avlTree, const void *e);
//Avl中插入关键码
AVLNODE *AvlInsert(AVLTREE *avlTree, const void *e);
//Avl中删除关键码所在节点，返回值：0成功，!0失败
int AvlRemove(AVLTREE *avlTree, void *e);
//Avl中删除关键码所在节点（无需深度删除关键码），返回值：0成功，!0失败
int AvlRemoveU(AVLTREE *avlTree, void *e);
#endif
```