```c
/* BTree.h */
#ifndef _BTREE_H
#define _BTREE_H
#include "OrderVector.h"

//0--代表比较相同，1--代表dataAddr<keyAddr，-1--代表dataAddr>keyAddr
typedef int  BTreeCmp(const void *keyAddr, const void *dataAddr);
typedef void BTreeFree(void *);
typedef void BTreeTraverseOp(void *, void *);

typedef struct bt_node
{
    struct bt_node *parent;
    VECTOR keyVector; //关键码向量
    VECTOR childVector; //孩子向量（其长度总比keyVector多一）
}BTREENODE;

typedef struct
{
    BTREENODE *root;
    BTREENODE *hot;//"命中"节点的父亲
    int size; //存放关键码总数
    int order; //B-树的阶次，至少为3，创建时指定，一般不能修改
    int keySize; //一个关键码所需字节数
    BTreeCmp *cmpFn;
    BTreeFree *freeFn;
}BTREE;

//BTree初始化
void BTreeNew(BTREE *bTree, int order, int keySize, BTreeCmp *cmpFn, BTreeFree
*freeFn);
//BTree销毁
void BTreeDispose(BTREE *bTree);
//BTree判空
int BTreeEmpty(BTREE *bTree);
//BTree规模
int BTreeSize(BTREE *bTree);
//BTree阶次
int BTreeOrder(BTREE *bTree);
//BTree中查找关键码
BTREENODE *BTreeSearch(BTREE *bTree, const void *e);
//BTree中插入关键码
int BTreeInsert(BTREE *bTree, const void *e);
//BTree中删除关键码
int BTreeRemove(BTREE *bTree, void *e);
//BTree层序遍历
void BTreeTravLevel(BTREE *bTree, BTreeTraverseOp *traverseOpFn, void *outData);
#endif
```