[**红黑树（数据结构大作业扩展版）**](http://www.cppblog.com/qywyh/articles/32740.html)

RBTree.h

http://www.cppblog.com/Images/OutliningIndicators/None.gif#ifndef RBTREE\_H  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#define RBTREE\_H  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifconst int RED = 0;  
http://www.cppblog.com/Images/OutliningIndicators/None.gifconst int BLACK = 1;  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifclass RBTreeNode http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifpublic:  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree关键字  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    KT key;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree记录  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RT rec;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree左儿子指针  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree右儿子指针  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree父节点指针  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*p;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree下一节点指针，该节点和下一节点有相同key值  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*nxt;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree上一节点指针，该节点和上一节点有相同key值  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*pre;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //节点颜色  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int color;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //节点构造函数  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    RBTreeNode(KT ckey, RT crec, int col) http://www.cppblog.com/Images/dot.gif{color=col; key=ckey; rec=crec; left=NULL; right=NULL; p=NULL; nxt=NULL; pre=NULL;};  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    RBTreeNode(int col) http://www.cppblog.com/Images/dot.gif{color = col; left=NULL; right=NULL; p=NULL; nxt=NULL; pre=NULL;};  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif};  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifclass RBTree http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifpublic:  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree哨兵指针  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*NIL;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree树根指针  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*root;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree构造函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTree();  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree析构函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    ~RBTree();  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree中序遍历函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void travel(RBTreeNode<KT, RT> \*v,int);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree左旋操作  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void leftRotate(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree右旋操作  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void rightRotate(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree插入函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void insert(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree插入调整函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void insertFixUp(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree查找x节点后继节点函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT>\* successor(RBTreeNode<KT, RT> \*x);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree查找以x为根的子树中的最小值节点函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT>\* getMin(RBTreeNode<KT, RT> \*x);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree删除节点函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void del(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree删除链表头节点函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void delFirst(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree删除链表头节点调整函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void delFirstFixUp(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree删除链表内部节点调整函数  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void delInter(RBTreeNode<KT, RT> \*z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    //RBTree查找节点(包括链表内部节点);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \* find(KT fkey, RT frec);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif};  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifRBTree<KT, RT>::RBTree() http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    NIL = new RBTreeNode<KT, RT>(BLACK);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    root = NIL;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifRBTree<KT, RT>::~RBTree() http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    delete NIL;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::travel(RBTreeNode<KT, RT> \*v, int tc) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*p;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (v == NIL) return ;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (tc>ans) ans=tc;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    printf("( ");  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    travel(v->left,tc+1);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    printf(" %d ", v->key);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    p = v->nxt;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while (p) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        printf(", %d ", p->key);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        p = p->nxt;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    travel(v->right,tc+1);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    printf(" )");  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::leftRotate(RBTreeNode<KT, RT> \*x) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*y = x->right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    x->right = y->left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    y->left->p = x;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    y->p = x->p;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (x->p == NIL) root = y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        if (x == x->p->left) x->p->left = y;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        else x->p->right = y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    y->left = x;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    x->p = y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::rightRotate(RBTreeNode<KT, RT> \*x) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*y = x->left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    x->left = y->right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    y->right->p = x;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    y->p = x->p;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (x->p == NIL) root = y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        if (x == x->p->left) x->p->left = y;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        else x->p->right = y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    y->right = x;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    x->p = y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::insert(RBTreeNode<KT, RT> \*z) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*y = NIL, \*x = root;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while (x != NIL) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        y = x;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        if (z->key < x->key) x = x->left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        else if (z->key > x->key) x = x->right;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            //加入该节点的链表  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            z->nxt = x->nxt;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            x->nxt = z;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            z->pre = x;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            if (z->nxt) z->nxt->pre = z;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            return ;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    z->p = y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    if (y == NIL) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        root = z;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        if (z->key < y->key) y->left = z;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        else y->right = z;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    z->left = NIL;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    z->right = NIL;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    z->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    insertFixUp(z);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::insertFixUp(RBTreeNode<KT, RT> \*z) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while (z->p->color == RED) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if (z->p == z->p->p->left) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            y = z->p->p->right;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            if (y->color == RED) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                y->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->p->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z = z->p->p;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif                if (z == z->p->right) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    z = z->p;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    leftRotate(z);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->p->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                rightRotate(z->p->p);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            y = z->p->p->left;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            if (y->color == RED) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                y->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->p->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z = z->p->p;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif                if (z == z->p->left) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    z = z->p;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    rightRotate(z);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                z->p->p->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                leftRotate(z->p->p);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    root->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifRBTreeNode<KT, RT>\* RBTree<KT, RT>::getMin(RBTreeNode<KT, RT> \*x) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    while (x->left != NIL) x = x->left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifRBTreeNode<KT, RT>\* RBTree<KT, RT>::successor(RBTreeNode<KT, RT> \*x) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*y;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (x->right != NIL) return getMin(x->right);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    y = x->p;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while (y != NIL && x == y->right) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        x = y;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        y = y->p;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::delFirst(RBTreeNode<KT, RT> \*z) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*x, \*y;   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (z->left == NIL || z->right == NIL) y = z;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    else y = successor(z);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (y->left != NIL) x = y->left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    else x = y->right;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    x->p = y->p;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (y->p == NIL) root = x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        if (y == y->p->left) y->p->left = x;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        else y->p->right = x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (y != z) z->key = y->key;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (y->color == BLACK) delFirstFixUp(x);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    delete y;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::delFirstFixUp(RBTreeNode<KT, RT> \*x) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*w;   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while (x != root && x->color == BLACK) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if (x == x->p->left) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            w = x->p->right;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            if (w->color == RED) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x->p->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                leftRotate(x->p);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w = x->p->right;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            if (w->left->color == BLACK && w->right->color == BLACK) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x = x->p;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif                if (w->right->color == BLACK) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    w->left->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    w->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    rightRotate(w);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    w = x->p->right;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->color = x->p->color;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x->p->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->right->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                leftRotate(x->p);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x = root;                      
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            w = x->p->left;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            if (w->color == RED) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x->p->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                rightRotate(x->p);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w = x->p->left;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            if (w->left->color == BLACK && w->right->color == BLACK) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x = x->p;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif            } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif                if (w->left->color == BLACK) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    w->right->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    w->color = RED;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    leftRotate(w);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                    w = x->p->left;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->color = x->p->color;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x->p->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                w->left->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                rightRotate(x->p);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif                x = root;                      
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    x->color = BLACK;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifRBTreeNode<KT, RT>\* RBTree<KT, RT>::find(KT fkey, RT frec) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*x = root;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    while (x) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        if (fkey < x->key) x = x->left;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        else if (fkey > x->key) x = x->right;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            while (x && x->rec != frec) x = x->nxt;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            return x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return x;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::delInter(RBTreeNode<KT, RT> \*z) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*pz = z->pre;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    pz->nxt = z->nxt;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    if (z->nxt) z->nxt->pre = pz;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    delete z;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.giftemplate <class KT, class RT>  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifvoid RBTree<KT, RT>::del(RBTreeNode<KT, RT> \*z) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<KT, RT> \*p;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    if (z->pre) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        //删除一个内部节点  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        delInter(z);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        //删除头节点  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        if (z->nxt) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            //删除该节点后链表不为空  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            p = z->nxt;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            z->key = p->key;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            z->rec = p->rec;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            delInter(p);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif        } else http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            //删除RBTree上的该节点  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif            delFirst(z);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#endif

test.cpp

http://www.cppblog.com/Images/OutliningIndicators/None.gif#include <iostream>  
http://www.cppblog.com/Images/OutliningIndicators/None.gif#include "RBTree.h"  
http://www.cppblog.com/Images/OutliningIndicators/None.gifusing namespace std;  
http://www.cppblog.com/Images/OutliningIndicators/None.gifint ans=0;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedBlock.gifint main() http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    int i;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTreeNode<int, int> \*h[100], \*tmp;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    RBTree<int, int> \*t;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    t = new RBTree<int, int>;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    for (i=0; i<10; i++) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        h[i] = new RBTreeNode<int, int>(10+i/2, 10000+i, RED);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        t->insert(h[i]);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        ans=0;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        t->travel(t->root,0);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        printf("\n");  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        printf("hight=%d\n", ans);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.cppblog.com/Images/OutliningIndicators/ContractedSubBlock.gif    for (i=0; i<10; i++) http://www.cppblog.com/Images/dot.gif{  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        tmp = t->find(10+i/2, 10000+i);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        if (tmp) t->del(tmp);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        ans=0;  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        t->travel(t->root,0);  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        printf("\n");  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        printf("hight=%d\n", ans);  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    system("pause");  
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    return 0;  
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}