Шаблон класса Tree

```
template<typename T> class Item {
     T key;  char b; Item<T> *left; Item<T> *right;
     Item<T> *A,*B,*C;
public:
     Item(const T& n=0, char c=0, Item<T>* l=0, Item<T>* r=0) { key = n; b=c; left = l; right = r; }
T&   get_key() { return key; }
Item*&    get_left() { return left; }
Item*&    get_right() { return right; }
void      ins(const T& K) {
     if(key == K) { h=0; return; }
     if(K < key)  { if(left) left->ins(K); else { left = new Item<T>(K); h++; }
               if(h) switch (b) {
                    case 1: h=0;
                    case 0: b--; break;
                    case -1: A = left;
                         if( A->b == -1) { B=new Item<T>; *B=*this; B->b=0; right = B; key = A->key;
                                   B->left = A->right; left = A->left; delete A; }
                         else { C = new Item<T>; *C = *this; B = A->right; A->right = B->left;
                              right = C; key=B->key; C->left = B->right;
                              C->b = (B->b == -1)?1:0;
                              A->b = (B->b == 1)?-1:0; delete B; }
                    b = 0; h = 0;
                    }         // switch
               }              // K < key
     else       { if(right) right->ins(K); else { right = new Item<T>(K); h++; }
               if(h) switch (b) {
                    case -1: h=0;
                    case 0: b++; break;
                    case 1: A = right;
                         if( A->b == 1) { B=new Item<T>; *B = *this; B->b=0; left = B; key = A->key;
                                   B->right = A->left; right = A->right; delete A; }
                         else {    C = new Item<T>; *C = *this; B = A->left; A->left = B->right;
                              left = C; key = B->key; C->right = B->left;
```

Шаблон класса Tree

```
                                C->b = (B->b == 1)?-1:0;
                                A->b = (B->b == -1)?1:0; delete B; }
                        b = 0; h = 0;
                        }       // end switch
                }               // end K > key
}
friend ostream& operator<<(ostream& a,const Item<T> b) {
static int k=0;
     k++; if(b.right!=NULL) a << *b.right; k--;
     for(int i=0;i<k;i++) a << '\t'; a << b.key << '(' << (int)b.b << ')' << '\n';
     k++; if(b.left!=NULL) a << *b.left; k--;
     return a; }
};

template<typename T> class Tree {
     Item<T> *top; T rab;
public:
Tree() { top = 0; }
void insert(T a[], int n) { int nl=n/2, nr=n-nl-1; if(n) insert(a[n/2]); else return;
         insert(a,nl); insert(&a[n/2+1],nr); }
void insert(const T& k) { if(top) top->ins(k); else top=new Item<T>(k),h++; }
friend ostream& operator<<(ostream& a,const Tree<T>& b) { return a << *b.top; }
};
```