0.106. aedsrc/aedcode/tree.h

```
#ifndef AED_TREE_H
#define AED_TREE_H

#include <cassert>
#include <iostream>
#include <cstddef>
#include <cstdlib>

namespace aed {

 //---:---<*>---:---<*>---:---<*>---:---<*>---:---<*>---:---<*>---:---<*>---:
 template<class T>
 class tree {
 public:
  class iterator;
 private:
  class cell {
   friend class tree;
   friend class iterator;
   T t;
   cell *right, *left_child;
   cell() : right(NULL), left_child(NULL) {}
  };
  cell *header;

  iterator tree_copy_aux(iterator nq,
                         tree<T> &TT,iterator nt) {
   nq = insert(nq,*nt);
   iterator
    ct = nt.lchild(),
    cq = nq.lchild();
   while (ct!=TT.end()) {
    cq = tree_copy_aux(cq,TT,ct);
    ct = ct.right();
    cq = cq.right();
   }
   return nq;
  }
 public:
  static int cell_count_m;
  static int cell_count() { return cell_count_m; }
  class iterator {
  private:
   friend class tree;
   cell *ptr,*prev,*father;
   iterator(cell *p,cell *prev_a,cell *f_a) : ptr(p),
    prev(prev_a), father(f_a) { }
```

```
48.     public:
49.       iterator(const iterator &q) {
50.         ptr = q.ptr;
51.         prev = q.prev;
52.         father = q.father;
53.       }
54.       T &operator*() { return ptr->t; }
55.       T *operator->() { return &ptr->t; }
56.       bool operator!=(iterator q) { return ptr!=q.ptr; }
57.       bool operator==(iterator q) { return ptr==q.ptr; }
58.       iterator() : ptr(NULL), prev(NULL), father(NULL) { }
59.
60.       iterator lchild() { return iterator(ptr->left_child,NULL,ptr); }
61.       iterator right() { return iterator(ptr->right,ptr,father); }
62.
63.       // Prefix:
64.       iterator operator++() {
65.         *this = right();
66.         return *this;
67.       }
68.       // Postfix:
69.       iterator operator++(int) {
70.         iterator q = *this;
71.         *this = right();
72.         return q;
73.       }
74.     };
75.
76.     tree() {
77.       header = new cell;
78.       cell_count_m++;
79.       header->right = NULL;
80.       header->left_child = NULL;
81.     }
82.     tree<T>(const tree<T> &TT) {
83.       if (&TT != this) {
84.         header = new cell;
85.         cell_count_m++;
86.         header->right = NULL;
87.         header->left_child = NULL;
88.         tree<T> &TTT = (tree<T> &) TT;
89.         if (TTT.begin()!=TTT.end())
90.           tree_copy_aux(begin(),TTT,TTT.begin());
91.       }
92.     }
93.     tree &operator=(tree<T> &TT) {
94.       if (this != &TT) {
95.         clear();
96.         tree_copy_aux(begin(),TT,TT.begin());
```

((document-version "tpl2-2019-63-g854be44c")
(document-date "Sat Nov 9 01:28:02 2019 -0300")

(processed-date "Sat Nov 9 01:35:47 2019 -0300"))

```
97.       }
98.       return *this;
99.     }
100.    ~tree() { clear(); delete header; cell_count_m--; }
101.    iterator insert(iterator p,T t) {
102.      assert(!(p.father==header && p.ptr));
103.      cell *c = new cell;
104.      cell_count_m++;
105.      c->right = p.ptr;
106.      c->t = t;
107.      p.ptr = c;
108.      if (p.prev) p.prev->right = c;
109.      else p.father->left_child = c;
110.      return p;
111.    }
112.    iterator erase(iterator p) {
113.      if(p==end()) return p;
114.      iterator c = p.lchild();
115.      while (c!=end()) c = erase(c);
116.      cell *q = p.ptr;
117.      p.ptr = p.ptr->right;
118.      if (p.prev) p.prev->right = p.ptr;
119.      else p.father->left_child = p.ptr;
120.      delete q;
121.      cell_count_m--;
122.      return p;
123.    }
124.
125.    iterator splice(iterator to,iterator from) {
126.      assert(!(to.father==header && to.ptr));
127.      if (from.ptr->right == to.ptr) return from;
128.      cell *c = from.ptr;
129.
130.      if (from.prev) from.prev->right = c->right;
131.      else from.father->left_child = c->right;
132.
133.      c->right = to.ptr;
134.      to.ptr = c;
135.      if (to.prev) to.prev->right = c;
136.      else to.father->left_child = c;
137.
138.      return to;
139.    }
140.    iterator find(T t) { return find(t,begin()); }
141.    iterator find(T t,iterator p) {
142.      if(p==end() || p.ptr->t == t) return p;
143.      iterator q,c = p.lchild();
144.      while (c!=end()) {
145.        q = find(t,c);
```

((document-version "tpl2-2019-63-g854be44c")
(document-date "Sat Nov 9 01:28:02 2019 -0300")

(processed-date "Sat Nov 9 01:35:47 2019 -0300"))

```
146.       if (q!=end()) return q;
147.       else c++;
148.      }
149.      return iterator();
150.    }
151.    void clear() { erase(begin()); }
152.    iterator begin() { return iterator(header->left_child,NULL,header); }
153.    iterator end() { return iterator(); }
154.
155.  };
156.
157.  template<class T>
158.  int tree<T>::cell_count_m = 0;
159.
160.  template<class T>
161.  void swap(tree<T> &T1, tree<T> &T2) { T1.swap(T2); }
162. }
163. #endif
```

0.107. aedsrc/aedcode/treebas.h

```
1.  class tree;
2.  class iterator_t;
3.
4.  //---:---<*>---:---<*>---:---<*>---:---<*>
5.  class cell {
6.    friend class tree;
7.    friend class iterator_t;
8.    elem_t elem;
9.    cell *right, *left_child;
10.   cell() : right(NULL), left_child(NULL) {}
11. };
12.
13. //---:---<*>---:---<*>---:---<*>---:---<*>
14. class iterator_t {
15. private:
16.   friend class tree;
17.   cell *ptr,*prev,*father;
18.   iterator_t(cell *p,cell *prev_a, cell *f_a)
19.     : ptr(p), prev(prev_a), father(f_a) { }
20. public:
21.   iterator_t(const iterator_t &q) {
22.     ptr = q.ptr;
23.     prev = q.prev;
24.     father = q.father;
25.   }
26.   bool operator!=(iterator_t q) { return ptr!=q.ptr; }
27.   bool operator==(iterator_t q) { return ptr==q.ptr; }
```