

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <limits.h>
4 #include "mapa.h"
5 #include "arvore.h"
6
7
8 struct smapa {
9     int chave;
10    int dados;
11    Mapa* esq;
12    Mapa* dir;
13 };
14
15
16 static Mapa *cria_no (int c, int d) {
17     Mapa *nn = (Mapa *)malloc(sizeof(Mapa));
18     if (nn!=NULL) {
19         nn->esq = nn->dir = NULL;
20         nn->chave = c;
21         nn->dados = d;
22     }
23     return nn;
24 }
25
26
27 Mapa* cria (void) {
28     //Mapa *new = (Mapa *)malloc(sizeof(Mapa));
29     //return new;
30     return NULL;
31 }
32 int busca (Mapa *m, int chave) {
33     while (m!=NULL) {
34         if (chave < m->chave)
35             m = m->esq;
36         else if (chave > m->chave)
37             m = m->dir;
38         else
39             return m->dados; /* achou */
40     }
41     return INT_MIN;
42 }
43
44 void destroi (Mapa *m) {
45     if (m==NULL) return;
46     destroi (m->esq);
47     destroi (m->dir);
48     free(m);
49 }
50
51 /*Mapa *insere (Mapa *m, int chave, int d) {
52     if (m==NULL)
53     }
54     m = cria_no(chave, d);
55     return m;
56 }
57 else
58 {
59     if(m->chave > chave)
60         return insere(m->esq,chave,d);
61     else
62         return insere(m->dir,chave,d);
63 }
64 }*/
65
66 Mapa *retira (Mapa *m, int chave) {
67     if (m==NULL)

```

```

68     return NULL;
69     return m;
70 }
71
72
73 Mapa* cria_raiz(int chave, int dados, Mapa* sae, Mapa* sad) {
74     Mapa *r = cria_no(chave, dados);
75     if (r != NULL) {
76         r->esq = sae;
77         r->dir = sad;
78     }
79     return r;
80 }
81
82 void mostra(Mapa* m) {
83     printf("[");
84     if (m != NULL) {
85         printf("<%d %d> ", m->chave, m->dados);
86         mostra(m->esq);
87         mostra(m->dir);
88     }
89     printf("] ");
90 }
91
92
93 int num_nos (Mapa *m) {
94     if(m == NULL)
95         return 0;
96     return 1 + num_nos(m->esq) + num_nos(m->dir);
97 }
98
99 int maior_chave (Mapa *m) {
100     if(m == NULL)
101         return INT_MIN;
102     else if(m->dir == NULL)
103         return m->chave;
104     else
105         return maior_chave(m->dir);
106 }
107
108 int num_maiores_que (Mapa *m, int n) {
109     int nos = 0;
110     if(m == NULL)
111         return 0;
112     else if(m->chave > n)
113     {
114         return 1 + num_maiores_que(m->dir,n) + num_maiores_que(m->esq,n);
115     }
116     else
117         return num_maiores_que(m->dir,n);
118     //return nos;
119 }
120

```