

## Atividade 9 - Árvore Binária de Busca (T1)

12 de novembro de 2025

### Problema A

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct no{
5     int chave;
6     struct no *e, *d;
7 } No;
8
9 No* insere(No *raiz, int chave){
10     if(!raiz){
11         No* novo_no = (No*)malloc(sizeof(No));
12
13         novo_no->chave = chave;
14         novo_no->e = NULL;
15         novo_no->d = NULL;
16         return novo_no;
17     }
18
19     if(chave > raiz->chave){
20         raiz->d = insere(raiz->d, chave);
21     } else if (chave < raiz->chave){
22         raiz->e = insere(raiz->e, chave);
23     }
24     return raiz;
25 }
26
27 void imprimePreOrdem(No* raiz){
28     if(!raiz) return;
29     printf(" %d", raiz->chave);
30     imprimePreOrdem(raiz->e);
31     imprimePreOrdem(raiz->d);
32 }
33
34 void imprimeEmOrdem(No* raiz){
35     if(!raiz) return;
36     imprimeEmOrdem(raiz->e);
37     printf(" %d", raiz->chave);
38     imprimeEmOrdem(raiz->d);
39 }
40
41 void imprimePosOrdem(No* raiz){
42     if(!raiz) return;
```

```
43
44     imprimePosOrdem(raiz->>e);
45     imprimePosOrdem(raiz->d);
46     printf(" %d", raiz->chave);
47 }
48
49 void impressao(No* raiz){
50     printf("Pre.."); imprimePreOrdem(raiz); printf("\n");
51     printf("In.."); imprimeEmOrdem(raiz); printf("\n");
52     printf("Post:"); imprimePosOrdem(raiz); printf("\n\n");
53 }
54
55 int main(){
56     int c; scanf("%d", &c);
57     for(int caso = 1; caso <= c; caso++){
58         printf("Case %d:\n", caso);
59         int n; scanf("%d", &n);
60         No* raiz = NULL;
61
62         for(int i = 0; i < n; i++){
63             int num; scanf("%d", &num);
64             raiz = insere(raiz, num);
65         }
66         impressao(raiz);
67     }
68     return 0;
69 }
```

## Problema B

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #include <string.h>
5
6 typedef long long ll;
7
8 typedef struct No{
9     char c;
10    struct No *e, *d;
11 } No;
12
13 bool sp = false;
14
15 No* insert(No* root, char carac){
16     if(!root){
17         No* no = (No*) malloc (sizeof(No));
18         no->e = NULL;
19         no->d = NULL;
20         no->c = carac;
21         return no;
22     }
23
24     if(carac > root->c){
25         root->d = insert(root->d, carac);
26     } else {
27         root->e = insert(root->e, carac);
28     }
29     return root;
30 }
31
32 bool search(No* root, char x){
33     if(!root) return false;
34     if(root->c == x) return true;
35
36     if(x > root->c){
37         return search(root->d, x);
38     } else return search(root->e, x);
39 }
40
41 void printInfix(No *root){
42     if(!root) return;
43
44     printInfix(root->e);
45
46     if(!sp) sp = true;
47     else printf(" ");
48
49     printf("%c", root->c);
50
51     printInfix(root->d);
52 }
53
54 void printPost(No *root){
55     if(!root) return;
56 }
```

```
57     printPost(root->e);
58     printPost(root->d);
59
60     if(!sp) sp = true;
61     else printf(" ");
62
63     printf("%c", root->c);
64 }
65
66 void printPre (No *root){
67     if(!root) return;
68
69     if(!sp) sp = true;
70     else printf(" ");
71
72     printf("%c", root->c);
73
74     printPre(root->e);
75     printPre(root->d);
76 }
77
78 int main(){
79     No* root = NULL;
80
81
82     char *o = (char*) malloc (sizeof(char) * 50);
83
84     while(scanf("%s\n", o) != EOF){
85         if(strcmp(o, "I") == 0){
86             char x; scanf("%c\n", &x);
87             root = insert(root, x);
88         } else if (strcmp(o, "INFIXA") == 0){
89             sp = false;
90             printInfix(root);
91             printf("\n");
92         } else if (strcmp(o, "PREFIXA") == 0){
93             sp = false;
94             printPre(root);
95             printf("\n");
96         } else if (strcmp(o, "POSFIXA") == 0){
97             sp = false;
98             printPost(root);
99             printf("\n");
100        } else {
101            char x; scanf("%c\n", &x);
102            if(search(root, x)) printf("%c existe\n", x);
103            else printf("%c nao existe\n", x);
104        }
105    }
106
107    return 0;
108 }
```