1.2/Trie.h

```c
#ifndef TRIE_H
#define TRIE_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

// Tamanho do alfabeto (# de símbolos)
#define TAMANHO_ALFABETO (26)

// Converte o caractere atual da chave em indice
// use apenas 'a' a 'z' em minúsculas
#define CHAR_PARA_INDICE(c) ((int)c - (int)'a')

// NO da trie
typedef struct TrieNode{
    struct TrieNode* filhos[TAMANHO_ALFABETO];
    // fimPalavra é verdadeiro se o NO representa
    // o final de uma palavra
    bool fimPalavra;
}NO;

// Retorna um novo NO da trie (inicializado como NULL)
NO* criaNO(){
    NO *p = NULL;
    p = (NO *)malloc(sizeof(NO));
    if (p){
        int i;
        p->fimPalavra = false;
        for (i = 0; i < TAMANHO_ALFABETO; i++)
            p->filhos[i] = NULL;
    }
    return p;
}

// Se não estiver presente, insere a chave na trie
// Se a chave eh um prefixo de um NO da trie, apenas marca o NO folha
void inserir(NO *raiz, const char *chave){
    int nivel;
    int comprimento = strlen(chave);
    int indice;
    NO *aux = raiz;
    for (nivel = 0; nivel < comprimento; nivel++){
        indice = CHAR_PARA_INDICE(chave[nivel]);
        if (!aux->filhos[indice])
            aux->filhos[indice] = criaNO();
        aux = aux->filhos[indice];
    }
    // marca o ultimo NO como folha
    aux->fimPalavra = true;
}

// Retorna verdadeiro se a chave estiver presente na trie, caso contrario, falso
bool buscar(NO *raiz, const char *chave){
    int nivel;
    int comprimento = strlen(chave);
```

```
58      int indice;
59      NO *aux = raiz;
60      for (nivel = 0; nivel < comprimento; nivel++){
61          indice = CHAR_PARA_INDICE(chave[nivel]);
62          if (!aux->filhos[indice])
63              return false;
64          aux = aux->filhos[indice];
65      }
66      return (aux->fimPalavra);
67  }
68
69  void pre_ordem(NO* raiz, int nivel){
70      if(raiz != NULL){
71          int i;
72          for(i=0; i<26; i++){
73              if(raiz->filhos[i] != NULL){
74                  printf("Nivel %d: ", nivel);
75                  printf("%c ", (char) 'a' + i);
76                  if(raiz->filhos[i]->fimPalavra) printf("*");
77                  printf("\n");
78                  pre_ordem(raiz->filhos[i], nivel+1);
79              }
80          }
81
82      }
83  }
84
85  void imprimeTrie(NO* raiz){
86      if(raiz == NULL) return;
87      pre_ordem(raiz, 0);
88      printf("\n");
89  }
90
91  #endif
```