



Si se definen las siguientes estructuras de datos para manejar un **Trie** de strings.

```
#define ALPHABET_SIZE 26

struct Trie_node {
    BOOL defineKey;          // true/false si define o no una clave
    struct Trie_node *children[ALPHABET_SIZE];
};

struct Trie {

    Trie();    // TODO
    ~Trie();   // TODO

    Trie_node *root;          // contenedor real de datos
    unsigned int numKeys;     // número de claves en el trie

    // inserta una nueva clave si no está presente o
    // lo marca con defineKey si ya existía como prefix de
    // otras claves
    void insertKey(const char *key);    // TODO

    // retorna verdadero/falso indicando si la clave existe.
    bool existKey(const char *key);    // TODO

    // Chequea la consistencia entre la cantidad de nodos que dice el
    // trie y la cantidad de nodos que definen key.
    // Retorna un bool indicando si hay o no consistencia.
    bool check();

    // Retorna la cantidad de claves que tienen un prefix
    // determinado.
    int numWordsWithPrefix(const char *prefix);

};
```

Notar que con esta representación no se guardan los caracteres (que en la figura anterior parecerían estar incluidos), estos están implícitos en la topología de la estructura: un nodo dado define el prefix de todos sus hijos, si **children[i]** es no vacío, **children[i]** representa al carácter **i** siguiendo al prefix.