

Cuarto taller obligatorio – Implementación de diccionario sobre trie

Condiciones de entrega

El taller se entrega teniendo un commit con los archivos `string_map.h` y `string_map.hpp` en el directorio `g4/taller` del repositorio de entregas individuales. El commit debe estar sincronizado en el repositorio remoto. Las correcciones del albotot sucederán todos los Lunes después de las 12hs del Domingo hasta el 15 de Julio.

Consigna

Se pide implementar la clase `string_map` que consiste en un diccionario con claves de tipo `string` y significados de tipo genérico, cuya interfaz será entregada por la cátedra en el archivo `string_map.h` adjunto. Dicha interfaz está basada en la clase `map` de la *Standard Library* de C++.

La estructura de representación interna de la clase `string_map` debe estar basada en la estructura *Trie* vista en la materia.

Interfaz de `string_map<T>`

1. `string_map<T>::string_map()` — Construye un diccionario vacío.
2. `string_map<T>::string_map(const string_map<T>& aCopiar)` — Construye un diccionario por copia.
3. `string_map& string_map<T>::operator=(const string_map& d)` — Operación de asignación.
4. `string_map<T>::~string_map()` — Destructor de la clase.
5. `T& string_map<T>::insert(const pair<string, T>& value_type)` — Definición del par clave/valor.
6. `int string_map<T>::count(const string& key) const` — Devuelve la cantidad de apariciones de la clave (0 o 1). Sirve para identificar si una clave está definida o no.
7. `const T& string_map<T>::at(const string& key) const` — Dada una clave, devuelve una referencia a su significado. Versión no modificable. *Precondición:* La clave está definida.
8. `T& string_map<T>::at(const string& key)` — Dada una clave, devuelve una referencia a su significado. Versión modificable. *Precondición:* La clave está definida.
9. `string_map<T>::erase(const string& key)` — Dada una clave, la borra del diccionario junto a su significado. *Precondición:* La clave está definida.
10. `int string_map<T>::size() const` — Devuelve cantidad de claves definidas.
11. `bool string_map<T>::empty() const` — Devuelve true si no hay ningún elemento en el diccionario.
12. `T& string_map<T>::operator[](const string& key)` — (Opcional) Acceso o definición de pares clave/valor.

La implementación que realicen **no debe perder memoria**. Recomendamos utilizar **valgrind** para testear si su implementación tiene *leaks* de memoria.

Para ejecutar los tests que incluyen funciones optativas, utilizar el target: `CORRERTESTS_EXT`.