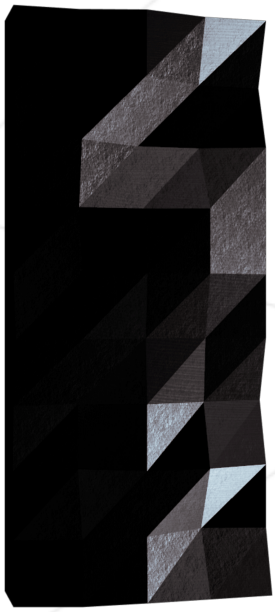

ft_containers

Para entender los contenedores de C++ de la forma más simple

Resumen: Cada contenedor disponible en C++ tiene un uso completamente distinto. Para que entiendas bien las diferencias, ¡vas a programarlos!

Índice general

Capítulo I

Reglas generales

- Si tienes headers desprotegidos, o funciones implementadas en headers (salvo si son templates)... Tendrás un 0 en el ejercicio.
- Los nombres de archivos son impuestos, se deben seguir al pie de la letra.
- Al igual que el nombre de los archivos, las clases, las funciones y los métodos también tienen nombres impuestos.
- Un pequeño recordatorio: estás programando en `C++`, no en `C`. Por lo tanto:
  - Las siguientes funciones están **prohibidas**, y su uso será recompensado con un `0`, sin lugar a interpretación: `*alloc`, `*printf`, y `free`.
  - Sin embargo, tienes permitido utilizar todo en la librería estándar. **Sin embargo**, sería más inteligente buscar la forma más C++ de las funciones que utilizarías en C; evita quedarte en lo que ya sabes, es un lenguaje nuevo.
  - Dado que tu objetivo es programar la librería STL, evidentemente no puedes utilizar los contenedores en sí.
- Para hacerlo más divertido, el uso de cualquier función o mecánica prohibida resultará en un `0`, sin necesidad de justificar nada.
- Ten en cuenta que, salvo que se escriba explícitamente, el uso de las palabras claves de `C++` `"using namespace"` y `"friend"` están prohibidas. Para estos dos casos en lugar de `0` tendrás un `-42`; sin justificación.
- Los archivos asociados a una clase tendrán como nombre el formato `NombreDeClase.hpp` y `NombreDeClase.cpp`, salvo que se diga lo contrario.
- Lee los ejemplos con cuidado: pueden tener requisitos que a simple vista pueden pasar desapercibidos en la descripción. Si consideras un ejercicio ambiguo, tienes que seguir aprendiendo `C++`.
- Como se te permite utilizar las herramientas de `C++` con las que has aprendido, no se te permite utilizar librerías externas. Esto incluye contenido de `C++11` o derivados (incluido `Boost`).
- Te recomendamos leer el proyecto **entero** antes de empezarlo. Hazlo, de verdad.

- El compilador que debes utilizar es `clang++`.
- Tu código debe compilarse con las siguientes flags: `-Wall -Werror -Wextra -std=c++98`.
- Cada archivo que incluyas debe ser capaz de incluirse independientemente del resto. Evidentemente si un archivo que incluyes depende de otro, debe incluirlo.
- Puedes programar siguiendo la norma de estilo que consideres oportuna, no se te impone ninguna en `C++`. Sin embargo, ten en cuenta que un código que tus compañeros no entiendan es algo que tampoco van a poder evaluar.
- **Importante:** no te evaluará un programa, sin embargo no seas perezoso; podrías perderte mucho de lo que este proyecto te ofrece.
- Siéntete libre en caso de que quieras separar tu código en múltiples archivos, no supone un problema (salvo si te evalúa un programa).
- Incluso si un ejercicio parece corto, merece la pena dedicarle algo de tiempo para estar completamente seguro de que entiendes lo que se espera de ti y que lo has hecho de la mejor forma posible.
- Por Odín, por Thor. ¡Utiliza el cerebro!

Capítulo II

Objetivos

- En este proyecto, tendrás que implementar los diversos contenedores de la STL de C++.
- Para cada contenedor, entrega los archivos nombrados propiamente según el nombre de su clase.
- El namespace será siempre `ft` y tus contenedores se probarán utilizando ft::<contenedor>.
- Debes respetar la estructura del contenedor de referencia. Sin embargo, si le falta parte de la forma canónico-ortodoxa... No la implementes.
- Como recordatorio, estás programando en `C++98`, así que las nuevas características de los contenedores **no tienen que ser implementadas**, pero todas las viejas (aunque estén deprecated) sí que se esperan.

Capítulo III

Parte obligatoria

- Debes implementar los siguientes contenedores y entregar los archivos necesarios `<contenedor>.hpp`.
- Deberás proporcionar también un `main.cpp` que pruebe todos los contenedores durante tus evaluaciones (ve más allá que los ejemplos).
- Deberás producir un binario solo con tus contenedores y otro con las mismas pruebas sobre los contenedores de la STL.
- Compara los resultados y los tiempos (como mucho, los tuyos pueden ir 20 veces más lento).
- Se espera que hagas las funciones miembro, funciones no miembro y sobrecargas.
- Respeta la nomenclatura, cuida los detalles.
- Debes utilizar std::allocator.
- Debes poder justificar la estructura inherente a cada uno de tus contenedores (utilizar un simple array para un mapa está mal).
- Si el contenedor tiene un sistema de iteración, debes implementarlo.
- `iterators_traits`, `reverse_iterator`, `enable_if`, `is_integral`, `std::pair`, `std::make_pair` y la comparación de igualdad/lexicográfica deben reimplementarse.
- Puedes utilizar https://www.cplusplus.com/ y https://cppreference.com como referencias.
- No puedes implementar más funciones públicas de las que ofrecen los contenedores estándar. Todo lo demás debe ser private o protected. Cada función y variable pública debe estar justificada.
- Para sobrecargas de no miembros, la palabra `friend` se permite. Cada uso de `friend` debe estar correctamente justificado y será comprobado durante la evaluación.

Debes entregar los siguientes contenedores y sus funciones asociadas:

- Vector
- Map
- Stack

`Para la implementación de los vector, no es obligatorio programar la especialización vector<bool>.`

Tu `stack` utilizará tu clase `vector` como contenedor subyacente por defecto, debe ser compatible con otros contenedores como el de la STL.
Los contenedores de la STL están prohibidos.
Tienes permitido utilizar la librería STD.

Capítulo IV

Parte bonus

Si has terminado la parte obligatoria, puedes intentar entregar este último contenedor como bonus:

- Set - pero esta vez un árbol Red and Black es obligatorio.