ft_containers

Контейнеры C++, простой режим

Резюме: все
стандартные контейнеры C++ имеют специфическое использование.
Чтобы убедиться, что вы их понимаете, давайте повторим их!

Версия: 4

Содержание

Глава I

Цели

В этом проекте вы реализуете несколько типов контейнеров из стандартной библиотеки шаблонов C++.

Вы должны взять структуру каждого стандартного контейнера в качестве эталона. Если в нем отсутствует часть православной канонической формы, не внедряйте ее.

Напоминаем, что вы должны соответствовать стандарту C++98, поэтому любые более поздние функции контейнеров НЕ ДОЛЖНЫ быть реализованы, но ожидается каждая функция C++98 (даже устаревшая).

Глава II

Основные правила

Компиляция

• Скомпилируйте свой код с помощью C++ и флагов -Wall -Wextra -Werror

• Ваш код все равно должен компилироваться, если вы добавите флаг -std=c++98.

Соглашения о форматировании и именовании

• Для каждого контейнера сдайте файлы классов с соответствующими именами.

• До свидания, Норминетт! Никакой стиль кодирования не применяется. Вы можете следить за своим любимым. Но имейте в виду, что код, который не могут понять ваши коллеги-оценщики, — это код, который они не могут оценить. Старайтесь писать чистый и читаемый код.

Разрешено/Запрещено

Вы больше не кодируете на C. Время C++! Следовательно:

• Вам разрешено использовать все из стандартной библиотеки. Таким образом, вместо того, чтобы придерживаться того, что вы уже знаете, было бы разумно использовать как можно больше C++-версий функций C, к которым вы привыкли.

• Однако вы не можете использовать никакую другую внешнюю библиотеку. Это означает, что C++11 (и производные формы) и библиотеки Boost запрещены. Также запрещены следующие функции: *printf(), *alloc() и free(). Если вы их используете, ваша оценка будет 0 и все.

Несколько требований к дизайну

• Утечка памяти происходит и в C++. Когда вы выделяете память, вы должны избегать утечки памяти.

• Любая реализация функции, помещенная в заголовочный файл (кроме шаблонов функций) означает 0 к упражнению.

• Вы должны иметь возможность использовать каждый из ваших заголовков независимо от других. Таким образом, они должны включать все необходимые им зависимости. Однако вы должны избежать проблемы двойного включения, добавив защиту включения. В противном случае ваша оценка будет 0.

Прочти меня

- Вы можете добавить несколько дополнительных файлов, если вам это нужно (т. е. разделить код) и организовать свою работу по своему усмотрению, пока вы сдаете обязательные файлы.

- Клянусь Одином, клянусь Тором! Используй свой мозг!!!

Поскольку ваша задача здесь — перекодировать контейнеры STL, вы, конечно , не можете использовать их для реализации своих.

Глава III

Обязательная часть

Реализуйте следующие контейнеры и включите необходимые файлы <container>.hpp:

- вектор

  Вам не нужно делать специализацию vector<bool>.

- карта
- куча

  Он будет использовать ваш векторный класс в качестве базового контейнера по умолчанию. Но он по-прежнему должен быть совместим с другими контейнерами, включая STL.

Вы можете пройти это задание без стека (80/100).
Но если вы хотите сделать бонусную часть, вам нужно реализовать 3 обязательных контейнера: вектор, карту и стек.

Вы также должны реализовать:

- iterators_traits
- реверс_итератор
- enable_if Да,

  это C++11, но вы сможете реализовать его в стиле C++98.

  Это задано, чтобы вы могли открыть для себя SFINAE.

- is_integral
- равные и/или лексикографические_сравнения
- станд::пара
- std::make_pair

III.1 Требования

- Пространство имен должно быть ft.
- Каждая внутренняя структура данных, используемая в ваших контейнерах, должна быть логичной и обоснованной (это означает, что использование простого массива для карты недопустимо).
- Вы не можете реализовать больше общедоступных функций, чем предлагается в стандартных контейнерах. Все остальное должно быть частным или защищенным. Каждая публичная функция или переменная должны быть обоснованы.
- Ожидаются все функции-члены, функции, не являющиеся членами, и перегрузки стандартных контейнеров.
- Вы должны следовать исходному именованию. Позаботьтесь о деталях.
- Если в контейнере есть система итераторов , вы должны реализовать ее.
- Вы должны использовать std::allocator.
- Для перегрузок, не являющихся членами, разрешено ключевое слово friend. Каждое использование друга должно быть обосновано и будет проверено во время оценки.
- Конечно, для реализации map::value_compare ключевое слово friend — это допустимый.

Вы можете использовать https://www.cplusplus.com/ и https://cppreference.com/ как ссылки.

III.2 Тестирование

- Вы также должны предоставить тесты, по крайней мере main.cpp, для вашей защиты. Вы должны пойти дальше, чем главное, приведенное в качестве примера!
- Вы должны создать два двоичных файла, которые запускают одни и те же тесты: один с вашими контейнерами только, а другой с контейнерами STL.
- Сравните выходные данные и производительность/время (в ваших контейнерах может быть до 20 контейнеров). раз медленнее).
- Протестируйте свои контейнеры с помощью: ft::<container>.

Файл main.cpp доступен для загрузки в интранет-проекте.

страница.

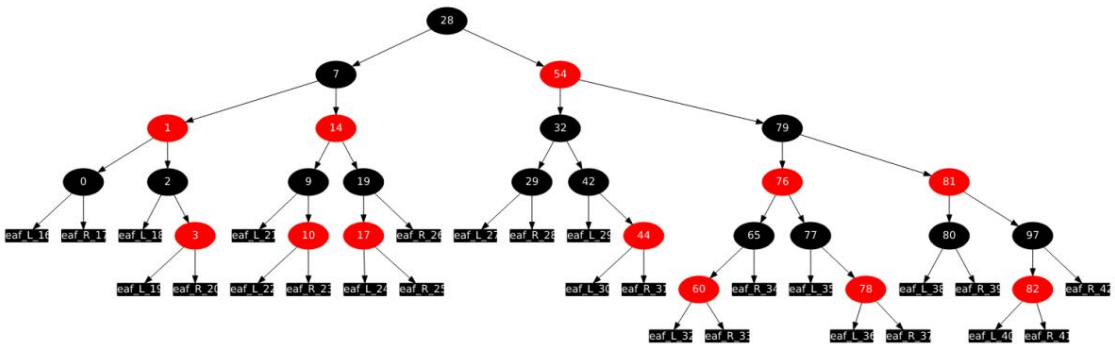
Глава IV

Бонусная часть

Вы получите дополнительные баллы, если реализуете последний контейнер:

- установлен

Но на этот раз красно-черное дерево обязательно .



Бонусная часть будет оцениваться только в том случае, если обязательная часть будет ИДЕАЛЬНОЙ. Идеальный означает, что обязательная часть была полностью выполнена и работает без сбоев. Если вы не выполнили ВСЕ обязательные требования, ваша бонусная часть вообще не будет оцениваться.

Глава V

Представление и экспертная оценка

Сдайте задание в своем репозитории Git, как обычно. Во время защиты будет оцениваться только работа внутри вашего репозитория. Не стесняйтесь перепроверять имена ваших файлов, чтобы убедиться, что они верны.