ft_контейнеры

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 контейнерных типов стандартной библиотеки шаблонов C++.

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

Напоминаю, что вы должны соответствовать стандарту C++98, поэтому любые более поздние возможности контейнеров **НЕ ДОЛЖНЫ** быть реализованы, но все возможности C++98 (даже устаревшие) ожидаются.

Глава II Общие правила

Компиляция

- Скомпилируйте ваш код с помощью c++ и флагов -Wall -Wextra -Werror
- Ваш код будет компилироваться, если вы добавите флаг -std=c++98

Форматирование и соглашения об именовании

- Для каждого контейнера создайте файлы классов с соответствующими именами.

- *До свидания, Норминет!* Стиль кодирования не навязывается. Вы можете следовать своему любимому стилю. Но помните, что код, который ваши коллеги-оценщики не могут понять, они не могут оценить. Делайте все возможное, чтобы писать чистый и читабельный код.

Разрешено/Запрещено

Вы больше не кодируете на C. Пора переходить на C++! Поэтому:

- 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++-шных версий функций языка C, к которым вы привыкли.

- Однако вы не можете использовать никакие другие внешние библиотеки. Это означает, что библиотеки C++11 (и производные формы) и 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 (за исключением шаблонов функций), означает 0 для упражнения.

- Вы должны иметь возможность использовать каждый из ваших заголовков

независимо от других. Таким образом, они должны включать все необходимые зависимости. Однако вы должны избегать проблемы двойного включения, добавляя **защитные элементы include**. В противном случае ваша оценка будет равна 0.

Читать

- Вы можете добавить несколько дополнительных файлов, если это необходимо (например, для разделения кода), и организовать свою работу по своему усмотрению, если вы сдаете обязательные файлы.
- Одином, Тором! Используйте свой мозг!!!

Поскольку ваша задача здесь - перекодировать контейнеры STL, вы, конечно, должны
не может использовать их для того, чтобы реализовать свои.

Глава III

Обязательная часть

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

- вектор
  Вам не нужно делать специализацию vector<bool>.
- карта
- стек
  Он будет использовать ваш класс вектора в качестве базового контейнера по умолчанию. Но он должен быть совместим с другими контейнерами, включая STL.

Вы можете сдать это задание без стека (80/100).
Но если вы хотите выполнить бонусную часть, вы должны реализовать 3 обязательных контейнера:vector,map и stack.

Вы также должны внедрять:

- черты итераторов
- обратный_ите
- ратор enable_if
  Да, это C++11, но вы сможете реализовать его на C++98. Это спрашивается для того, чтобы вы могли открыть для себя SFINAE.
- является_интегралом
- равный и/или лексикографическое_сравнение
- std::pair
- 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 map не подходит).
  - Вы не можете реализовать больше публичных функций, чем те, которые предлагаются в стандартных контейнерах. Все остальное должно быть приватным или защищенным. Каждая публичная функция или переменная должна быть **обоснована**.
  - Ожидаются все функции-члены, функции-нечлены и перегрузки стандартных контейнеров.
- Вы должны следовать оригинальному названию. Позаботьтесь о деталях.
- Если контейнер имеет систему **итераторов**, вы должны реализовать ее.
- Вы должны использовать std::allocator.
  - Для перегрузок, не являющихся членами, разрешается использовать ключевое слово friend. Каждое использование friend должны быть обоснованы и будут проверяться во время оценки.
  - Конечно, для реализации map::value_compare ключевое слово friend разрешено.

Вы можете использовать https://www.cplusplus.com/ и https://cppreference.com/ в качестве ссылок.

III.2 Тестирование

- Вы также должны предоставить тесты, по крайней мере, main.cpp, для вашей защиты. Вы должны пойти дальше, чем приведенный в качестве примера main!
  - Вы должны создать два двоичных файла, выполняющих одни и те же тесты: один только с вашими контейнерами, а другой - с контейнерами STL.
  - Сравните **выходы** и **производительность / время** (ваши контейнеры могут быть до 20 раз медленнее).
- Проверьте свои контейнеры с помощью: ft::<контейнер>.

Файл main.cpp доступен для загрузки на странице проекта в интрасети.

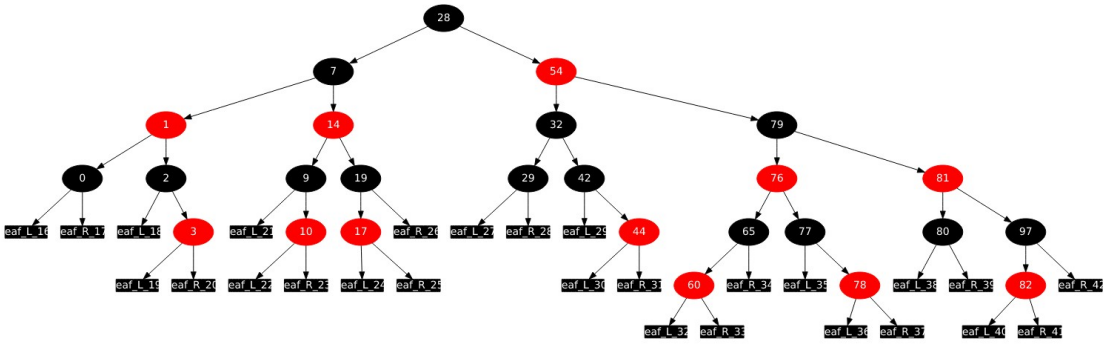
Глава IV

Бонусная часть

Вы получите дополнительные очки, если внедрите последний контейнер:

- установить

Но на этот раз **дерево красно-черного** цвета **обязательно.**



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

Глава V

Представление и экспертная оценка

Сдайте задание в свой Git-репозиторий, как обычно. Во время защиты будет оцениваться только та работа, которая находится в вашем репозитории. Не стесняйтесь дважды проверять имена файлов, чтобы убедиться в их правильности.