).
2

Глава II

Общие 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

•
Вы должны включить Makefile, который будет компилировать ваши исходные файлы. Он не должен
перестраиваться.

•
Ваш Makefile должен, по крайней мере, содержать правила:
$(ИМЯ), all, clean, fclean и re.

Соглашения о форматировании и именовании

•
Для каждого контейнера включите файлы классов с соответствующими именами.

•
Прощай, Норма! Никакой стиль кодирования не применяется принудительно. Вы можете следить за своим любимым.
Но имейте в виду, что код, который ваши коллеги-оценщики не могут понять, - это код, который они
не могут оценить. Сделайте все возможное, чтобы написать чистый и читаемый код.

Разрешено / Запрещено

Вы больше не пишете на языке Си. 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 ++-версий функций C, к которым вы привыкли.

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

Несколько требований к дизайну

•
Утечка памяти происходит и в C ++. Когда вы выделяете память, вы должны избегать
утечек памяти.
3
ft_контейнеры
Контейнеры C ++, простой режим

•
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 им зависимости. Однако вы должны избежать
проблемы двойного включения, добавив **защитные элементы включения**. В противном случае ваша оценка будет
равна 0.
Прочти меня
•
Вы можете добавить несколько дополнительных файлов, если вам нужно (например, разделить свой код), и
организовать свою работу по своему усмотрению, если вы включите обязательные файлы.
•
Клянусь Одином, клянусь Тором! Используй свой мозг!!!
Поскольку ваша задача здесь состоит в том, чтобы перекодировать контейнеры STL, вы, конечно
, не можете использовать их для реализации своих.
4

Глава III

Обязательная часть

Реализуйте следующие контейнеры и включите необходимые файлы <container>.hpp с помощью Makefile:
•
вектор
Вам не нужно выполнять специализацию vector<bool>.
•
карта
•
stack
It будет использовать ваш векторный класс в качестве базового контейнера по умолчанию. Но он все равно должен быть
совместим с другими контейнерами, включая контейнеры STL.
Вы можете передать это задание без стека (80/100).
Но если вы хотите выполнить бонусную часть, вы должны реализовать 3 обязательных контейнера:
вектор, карта и стек.
Вы также должны реализовать:
•
std::iterator_traits
•
std::reverse_iterator std::reverse_iterator
•
std::enable_if
Да, это C ++ 11, но вы сможете реализовать его на C ++ 98.
Это задано для того, чтобы вы могли открыть для себя SFINAE.
•
std::is_integral

•
std::равно и / или std::lexicographical_compare
•
std::пара
•
std::make_pair
5
ft_контейнеры
Контейнеры C++, требования к простому режиму

III.1

•
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 недопустимо).
•
Вы не можете реализовать больше общедоступных функций, чем те, которые предлагаются в стандартных
контейнерах. Все остальное должно быть закрытым или защищенным. Каждая общедоступная функция или
переменная должна быть **обоснована**.
•
Ожидаются все функции-члены, функции, не являющиеся членами, и перегрузки стандартных контейнеров.
•
Вы должны следовать первоначальному названию. Позаботьтесь о деталях.
•
Если контейнер имеет систему итераторов, вы должны ее реализовать.
•
Вы должны использовать std::allocator .
•
Для перегрузок, не связанных с участниками, ключевое слово friend разрешено. Каждое использование друга
должно быть обосновано и будет проверяться во время оценки.
•
Конечно, для реализации std::map::value_compare разрешено ключевое
слово friend .
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 основного, приведенного в качестве примера!
•

Вы должны создать два двоичных файла, которые выполняют одни и те же тесты: один только с вашими контейнерами
, а другой - с контейнерами STL.
•
Сравните **выходные** данные и **производительность / время** (ваши контейнеры могут работать в 20
раз медленнее).
•
Проверьте свои контейнеры с помощью: ft::<контейнер>.
A main.cpp файл доступен для загрузки на
странице проекта в интрасети.
6

Глава IV

Бонусная часть

Вы получите дополнительные очки, если реализуете один последний контейнер:
•
установить
Но на этот раз **Красно-Черное дерево обязательно**.
Бонусная часть будет начислена только в том случае, если обязательная часть
Идеальный. Perfect означает, что обязательная часть была выполнена полностью
и работает без сбоев.
Если вы не выполнили ВСЕ
обязательные требования, ваша бонусная часть вообще не будет оцениваться.
7

Глава V

Представление и экспертная оценка

Включите свое задание в свой репозиторий Git, как обычно.
Во время защиты будет оцениваться только работа внутри вашего репозитория.
Не стесняйтесь дважды проверять
имена ваших файлов, чтобы убедиться в их правильности.
8