allocator

Условие

Реализовать свой аллокатор памяти, который позволит добиться аналогичного `reserve(size_t)` размещения элементов для контейнера `std::map`. Аллокатор должен параметризоваться количеством выделяемых за один раз элементов. Освобождение конкретного элемента не предполагается - аллокатор должен освобождать всю память при освобождении всех элементов.

Реализовать свой контейнер, который по аналогии с контейнерами stl параметризуется аллокатором. Контейнер должен иметь две возможности - добавить новый элемент и обойти контейнер в одном направлении. Обязательно наличие конструктора и деструктора.

Прикладной код должен содержать следующие вызовы:

```
struct hard {
    int fa;
    int fi;
    hard(int fa, int fi) : fa(fa), fi(fi) {};
};
```

- создание экземпляра `std::map<int, hard>`
- заполнение 10 элементами, где ключ это число от 0 до 9, а значение - соответствующие значению ключа факториал и число Фибоначчи
- создание экземпляра `std::map<int, hard>` с новым аллокатором ограниченным 10 элементами
- заполнение 10 элементами, где ключ это число от 0 до 9, а значение - аналогичные первому контейнеру
- вывод на экран всех значений (ключ и значения разделены пробелами) хранящихся в контейнере
- создание экземпляра своего контейнера для хранения `hard`
- заполнение 10 значениями аналогичными первому контейнеру
- создание экземпляра своего контейнера для хранения `hard` с новым аллокатором ограниченным 10 элементами
- заполнение 10 значениями аналогичными первому контейнеру
- вывод на экран всех значений хранящихся в контейнере

Результат

- пакет `allocator` содержащий исполняемый файл `allocator` опубликован на bintray
- отправлена на проверку ссылка на страницу решения (github)

Проверка

Задание считается выполненным успешно, если после просмотра кода, подключения репозитория, установки пакета и запуска бинарного файла командой:

```
$ allocator
```

на экран выведется содержимое контейнеров с новым аллокатором. Команда

```
$ ltrace -e malloc -e free allocator
```

покажет снижение количества операций выделения памяти.

Оценка

баллы	за что
5	Успешно прошла проверка.
3	`hard(const hard &)` и `hard(hard &&) noexcept` помечены как `delete` и реализована идеальная передача.
1	Контейнер имеет публичный и приватный интерфейсы.
2	Вывод данных контейнера через for-loop.
2	Реализованы копирующий и перемещающий конструкторы при совпадении типа аллокатора. В коде есть пример.
2	Реализованы копирующий и перемещающий конструкторы при отличающихся типах аллокатора. В коде есть пример.

баллы	за что
4	Аллокатор может расширяться с шагом 10. В коде все циклы увеличены на 1, чтобы спровоцировать расширение.