Задание 3

Реализовать свой аллокатор памяти, который позволит выполнять операцию резервирования памяти. Далее использовать этот аллокатор с контейнером `std::map`. Аллокатор должен параметризоваться количеством выделяемых за раз элементов. Освобождение конкретного элемента не предполагается - аллокатор должен освобождать всю память самостоятельно. Аллокатор работает с фиксированным количеством элементов. Попытку выделить большее число элементов считать ошибкой.

Опционально реализовать расширяемость аллокатора. При попытке выделить число элементов, которое превышает текущее зарезервированное количество, аллокатор расширяет зарезервированную память.

Опционально реализовать поэлементное освобождение.

Цель такого аллокатора – снизить количество операций выделения памяти.

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.

Опционально реализовать совместимость с контейнерами stl – итераторы, вспомогательные методы size, empty и т.д.

Цель реализации своего контейнера – попробовать использовать std::allocator, а также свой аллокатор.

Прикладной код должен содержать следующие вызовы:

- создание экземпляра `std::map<int, int>`

- заполнение 10 элементами, где ключ - это число от 0 до 9, а значение - факториал ключа

- создание экземпляра `std::map<int, int>` с новым аллокатором, ограниченным 10 элементами

- заполнение 10 элементами, где ключ - это число от 0 до 9, а значение - факториал ключа

- вывод на экран всех значений (ключ и значение разделены пробелом) хранящихся в контейнере

- создание экземпляра своего контейнера для хранения значений типа `int`

- заполнение 10 элементами от 0 до 9

- создание экземпляра своего контейнера для хранения значений типа `int` с новым аллокатором, ограниченным 10 элементами

- заполнение 10 элементами от 0 до 9

- вывод на экран всех значений, хранящихся в контейнере