02 Рандомизированная очередь и дек

Jump to bottom

Alexander Morozov edited this page 12 days ago · 9 revisions

Ссылка на задание: https://classroom.github.com/a/VY9mYgZM

Задание

Необходимо реализовать шаблонные структуры данных: рандомизированную очередь (randomized_queue) и дек (deque). В задании разрешается использовать следующие структуры данных: std::list, std::vector, std::array. Остальную часть, не относящуюся к структурам данных, можно использовать по своему усмотрению.

Рандомизированная очередь - это коллекция, которая предоставляет доступ к своим элементам в случайном порядке. Таким образом, каждый отдельный "взгляд" на эту коллекцию даёт случайную, независимую от других, перестановку элементов. Например, если взять два итератора на начало коллекции, а затем каждым пройти по всем элементам до конца, то эти два прохода дадут две разных, независимых друг от друга, перестановки.

Пример: исходная коллекция = 1, 2, 3, 4, 5, 6

Перестановки:

6, 2, 4, 1, 5, 3

5, 3, 2, 6, 4, 1

2, 6, 4, 5, 3, 1

Подобный код должен работать:

```
randomized_queue<int> q;
for (int i = 0; i < 5; ++i) {
    q.enqueue(i);
}
auto b1 = q.begin();
auto e1 = q.end();
auto b2 = q.begin();
auto e2 = q.end();

std::vector<int> v11, v12;
std::copy(b1, e1, std::back_inserter(v11));
std::copy(b1, e1, std::back_inserter(v12));
assert(v11 == v12); // Два прохода одним итератором дают одинаковую последовательность

std::vector<int> v21, v22;
std::copy(b2, e2, std::back_inserter(v21));
std::copy(b2, e2, std::back_inserter(v22));
assert(v21 == v22); // Два прохода одним итератором дают одинаковую последовательность

assert(v11 != v21); // С высокой степенью вероятности, два разных итератора задают разные последовательности

// взятие итераторов не повлияло на очередь
while (!q.empty()) {
    std::cout << q.dequeue() << ' ';
}
```

Требования к дек

Необходимо создать шаблонную реализацию deque, которая удовлетворяет следующим условиям:

- конструктор без параметров
- метод empty, который отвечает на вопрос о пустоте структуры данных
- метод size возвращает количество элементов в структуре данных
- должны быть реализованы итераторы

- методы push_front, push_back - добавляют в начало/конец структуры данных элемент
- методы front, back - позволяют посмотреть на элемент в начале и конце
- методы pop_front, pop_back удаляют элемент из дека, с начала и с конца соотвественно.

Требования к рандомизированной очереди

Необходимо создать шаблонную реализацию randomized_queue, которая удовлетворяет следующим условиям:

- конструктор без параметров
- метод empty, который отвечает на вопрос о пустоте структуры данных
- метод size возвращает количество элементов в структуре данных
- должны быть реализованы итераторы (в том числе, позволяющие модификацию элементов очереди)
- методы enqueue - добавляют в структуру данных элемент
- методы sample - позволяют посмотреть случайный элемент, но при этом не удаляет его
- методы dequeue - возвращает случайный элемент и удаляет его из из дека

Клиентская программа subset

Нужно разработать утилиту с названием subset, которая принимает список строк и выдаёт k из них с равномерным распределением. Для этого нужно использовать разработанные структуры данных. При этом "строка" - это произвольная последовательность печатных символов, ограниченная символом перевода строки (`\n`).

In: `printf '%s\n' A B C D E F G H I | subset 3` Out: `C G A`

PS

1. Для сигнатур методов нужно выбрать правильные модификаторы доступа
2. Работа должно удовлетворять общим требованиям Требования к выполнению домашних заданий

HINT

1. Итераторы можно прокинуть из stl контейнеров
2. Если разрабатываете свой итератор, то он должен быть stl-like, а тип итератора остаётся на ваш выбор (но этот выбор должен быть разумным)

Clone this wiki locally

```
https://github.com/itiviti-cpp/wiki.wiki.git
```