Строки. Ввод/вывод

immutable string, shared_ptr, ostream

Цели и задачи занятия

- Вспомнить, что std::string - класс изменяемых строк
- Придумать интерфейс неизменяемой C++ строки
- Реализовать неизменяемую C++ строку с бесплатным копированием и другими оптимизациями

Изменяемые строки std::string

```
std::string str(“foo bar”);

str[0] = ‘o’;

assert(strcmp(str.c_str(), “ooo bar”) == 0);
```

Чем неизменяемые строки хуже/лучше изменяемых?

Тесты и стартовый код

https://drive.google.com/folderview?id=0B-KXDlig5kwVbWpxOFprZ1hQdlE&usp=sharing

Задача 1

- Реализуйте класс неизменяемой строки imstring:
  - С методами: begin(), end(), size(), c_str()
  - С операторами: доступ по индексу, вывод в поток, конкатенация строк
  - Семантика методов и операторов повторяет семантику в std::string
- Запретите присваивание imstring
- imstring может неявно создаваться от const char* и копироваться от const imstring&, imstring&&
- Во всех заданиях imstring поддерживает строгую гарантию безопасности исключений

Задача 2

Сделайте копирование условно бесплатным - при копировании imstring не происходит аллокации нового буфера и копирования содержимого исходной строки

Задача 3

Сделайте ленивую конкатенацию:

- При конкатенации 2-х imstring не происходит выделения нового буфера и копирования символов. Сложность от суммарной длины конкатенируемых строк: O(1)
- Буфер создается только при вызове методов, требующих его создания
- Буфер кешируется - не создается при повторных вызовах этих методов
- Метод size() и оператор << у сконкатенированной строки не должны создавать новый буфер

Задача 4 (ninja level)

Реализуйте методы begin() и end() так, чтобы они не создавали новый буфер, при вызове у сконкатенированной строки.

Итераторы, возвращаемые из begin() и end(), должны быть как минимум forward