

Лабораторная работа 2

Введение в unit тестирование.

Задание 1. (1 балл)

В этой задаче вам будет дано правильное решение условной задачи «Автобусные остановки», целиком содержащееся внутри функции `main`, файла `bus.cpp` Вам надо будет выполнить декомпозицию этого решения на заранее заданные блоки так, чтобы получившаяся программа так же корректно решала задачу. Условие задачи «Автобусные остановки — 1» приведено ниже.

Автобусные остановки. Реализуйте систему хранения автобусных маршрутов. Вам нужно обрабатывать следующие запросы:

- **NEW_BUS** *bus stop_count stop1 stop2 ...* — добавить маршрут автобуса с названием *bus* и *stop_count* остановками с названиями *stop1, stop2, ...*
- **BUSES_FOR_STOP** *stop* — вывести названия всех маршрутов автобуса, проходящих через остановку *stop*.
- **STOPS_FOR_BUS** *bus* — вывести названия всех остановок маршрута *bus* со списком автобусов, на которые можно пересесть на каждой из остановок.
- **ALL_BUSES** — вывести список всех маршрутов с остановками.

Формат ввода

В первой строке ввода содержится количество запросов Q , затем в Q строках следуют описания запросов.

Гарантируется, что все названия маршрутов и остановок состоят лишь из латинских букв, цифр и знаков подчёркивания.

Для каждого запроса **NEW_BUS** *bus stop_count stop1 stop2 ...* гарантируется, что маршрут *bus* отсутствует, количество остановок больше 0, а после числа *stop_count* следует именно такое количество названий остановок, причём все названия в каждом списке различны.

Формат вывода

Для каждого запроса, кроме **NEW_BUS**, выведите соответствующий ответ на него:

- На запрос **BUSES_FOR_STOP** *stop* выведите через пробел список автобусов, проезжающих через эту остановку, в том порядке, в котором они создавались командами **NEW_BUS**. Если остановка *stop* не существует, выведите **No stop**.

- На запрос **STOPS_FOR_BUS** *bus* выведите описания остановок маршрута *bus* в отдельных строках в том порядке, в котором они были заданы в соответствующей команде **NEW_BUS**. Описание каждой остановки *stop* должно иметь вид **Stop stop: bus1 bus2 ...**, где *bus1 bus2 ...* — список автобусов, проезжающих через остановку *stop*, в порядке, в котором они создавались командами **NEW_BUS**, за исключением исходного маршрута *bus*. Если через остановку *stop* не проезжает ни один автобус, кроме *bus*, вместо списка автобусов для неё выведите **no interchange**. Если маршрут *bus* не существует, выведите **No bus**.
- На запрос **ALL_BUSES** выведите описания всех автобусов в алфавитном порядке. Описание каждого маршрута *bus* должно иметь вид **Bus bus: stop1 stop2 ...**, где *stop1 stop2 ...* — список остановок автобуса *bus* в порядке, в котором они были заданы в соответствующей команде **NEW_BUS**. Если автобусы отсутствуют, выведите **No buses**.

Решать данную задачу не надо!!! У вас уже есть готовое решение bus.cpp

Кроме того, вам дан файл **starter-bus.cpp** который содержит заготовки классов и функций. Не меняя функцию `main`, вам надо реализовать эти классы и функции так, чтобы получившаяся программа решала задачу «Автобусные остановки».

Пример

Ввод

```
1 10
2 ALL_BUSES
3 BUSES_FOR_STOP Marushkino
4 STOPS_FOR_BUS 32K
5 NEW_BUS 32 3 Tolstopaltsevo Marushkino Vnukovo
6 NEW_BUS 32K 6 Tolstopaltsevo Marushkino Vnukovo Peredelkino Solntsevo Skolkovo
7 BUSES_FOR_STOP Vnukovo
8 NEW_BUS 950 6 Kokoshkino Marushkino Vnukovo Peredelkino Solntsevo Troparyovo
9 NEW_BUS 272 4 Vnukovo Moskovsky Rumyantsevo Troparyovo
10 STOPS_FOR_BUS 272
11 ALL_BUSES
```

Вывод

```
1 No buses
2 No stop
3 No bus
4 32 32K
5 Stop Vnukovo: 32 32K 950
6 Stop Moskovsky: no interchange
7 Stop Rumyantsevo: no interchange
8 Stop Tropar'yovo: 950
9 Bus 272: Vnukovo Moskovsky Rumyantsevo Tropar'yovo
10 Bus 32: Tolstopaltsevo Marushkino Vnukovo
11 Bus 32K: Tolstopaltsevo Marushkino Vnukovo Peredelkino Solntsevo Skolkovo
12 Bus 950: Kokoshkino Marushkino Vnukovo Peredelkino Solntsevo Tropar'yovo
```

Задание 2. (1 балл)

Тесты для функции `GetDistinctRealRootCount`.

Функция `int GetDistinctRealRootCount(double a, double b, double c);` возвращает количество уникальных действительных корней уравнения $ax^2 + bx + c = 0$. Разработайте набор юнит-тестов для проверки корректности реализации этой функции.

Начать работу вы можете с шаблона `task2.cpp`, который содержит фреймворк юнит-тест и заготовку функции `GetDistinctRealRootCount`.

Задание 3. (1 балл)

Вам необходимо разработать набор тестов для задачи «Имена и фамилии - 1». В ней надо было разработать класс `Person`, поддерживающий историю изменений человеком своих фамилии и имени. В данной задаче вам надо разработать юнит-тесты на реализацию класса `Person`. При разработке тестов учитывайте ограничения, которые накладывает на класс `Person` условие задачи «Имена и фамилии - 1».

Начать работу вы можете с шаблона `task3.cpp`, который содержит фреймворк юнит-тестов и заготовку класса.

Условие задачи «Имена и фамилии — 1»

Реализуйте класс для человека, поддерживающий историю изменений человеком своих фамилии и имени.

```

1 class Person {
2 public:
3     void ChangeFirstName(int year, const string& first_name) {
4         // добавить факт изменения имени на first_name в год year
5     }
6     void ChangeLastName(int year, const string& last_name) {
7         // добавить факт изменения фамилии на last_name в год year
8     }
9     string GetFullName(int year) {
10        // получить имя и фамилию по состоянию на конец года year
11    }
12 private:
13     // приватные поля
14 };
15

```

Считайте, что в каждый год может произойти не более одного изменения фамилии и не более одного изменения имени. При этом с течением времени могут открываться всё новые факты из прошлого человека, поэтому годá в последовательных вызовах методов `ChangeLastName` и `ChangeFirstName` не обязаны возрастать.

Гарантируется, что все имена и фамилии непусты.

Строка, возвращаемая методом `GetFullName`, должна содержать разделённые одним пробелом имя и фамилию человека по состоянию на конец данного года.

- Если к данному году не случилось ни одного изменения фамилии и имени, верните строку **"Incognito"**.
- Если к данному году случилось изменение фамилии, но не было ни одного изменения имени, верните **"last_name with unknown first name"**.
- Если к данному году случилось изменение имени, но не было ни одного изменения фамилии, верните **"first_name with unknown last name"**.

Задание 4. (1 балл)

Класс `Rational` представляет собой рациональное число и имеет следующий интерфейс

```

1 class Rational {
2 public:
3     Rational();
4     Rational(int numerator, int denominator);
5
6     int Numerator() const;
7     int Denominator() const;
8 };
9

```

Список требований, предъявляемых к реализации интерфейса класса `Rational`:

1. Конструктор по умолчанию должен создавать дробь с числителем 0 и знаменателем 1.
2. При конструировании объекта класса Rational с параметрами p и q должно выполняться сокращение дроби p/q.
3. Если дробь p/q отрицательная, то объект Rational(p, q) должен иметь отрицательный числитель и положительный знаменатель.
4. Если дробь p/q положительная, то объект Rational(p, q) должен иметь положительные числитель и знаменатель (обратите внимание на случай Rational(-2, -3)).
5. Если числитель дроби равен нулю, то знаменатель должен быть равен 1.

Разработайте набор юнит-тестов, которые будут проверять корректность реализации класса Rational.

Начать работу вы можете с шаблона task4.cpp , который содержит наш фреймворк юнит-тестов и заготовку класса Rational.

Задание 5. (1 балл)

В этой задаче вам нужно разработать набор юнит-тестов для функции

```
1 bool IsPalindrom(const string& s);
```

Эта функция проверяет, является ли строка s палиндромом. Палиндром — это слово или фраза, которые одинаково читаются слева направо и справа налево. Примеры палиндромов: **madam, level, wasitacaroracatisaw**

Разработайте набор юнит-тестов, который будет принимать правильные реализации функции IsPalindrom и отвергать неправильные. При этом учитывайте, что **правильная** реализация функции:

- считает пустую строку палиндромом;
- считает строку из одного символа палиндромом;
- осуществляет обычное сравнение символов на равенство, не игнорируя никакие символы, в том числе пробельные.

При разработке тестов подумайте, какие ошибки можно допустить при реализации функции IsPalindrom. Примеры ошибок:

- игнорируется первый или последний символ;
- сравнение соответствующих символов завершается не в середине строки, а раньше;

- игнорируются пробелы

Начать работу вы можете с шаблона, который содержит наш фреймворк юнит-тестов и заготовку функции `IsPalindrom`.