Тема: шаблоны

Вариант: 3.1.1

Задача: Реализовать шаблонный класс:

```
template <typename K, typename V>
class HashMap {
    ...
}
```

реализующий хеш-таблицу - контейнер для хранения пар <"ключ", "значение">, где ключи имеют тип K, а значения - тип V. При этом, все ключи *уникальные*, т.е. не существует двух пар с одинаковыми ключами. Пары <"ключ", "значение"> далее будем также называть *элементами* коллекции.

Для данной коллекции реализовать следующие операции:

1. Поиск элемента в коллекции по заданному ключу.

2. Добавление нового элемента с заданными ключом и значением в коллекцию.

3. Удаление из коллекции элемента с заданным ключом.

Временная сложность в среднем данных операций должна быть $O(1)$ (константной).

Замечания по реализации:

1. Для реализации хеш-функции предлагается использовать шаблонный класс `std::hash` из стандартной библиотеки

2. Для разрешения коллизий требуется использовать метод цепочек

3. Для достижения требуемой эффективности операций необходимо поддерживать разреженность хеш-таблицы. При достижении заданного в конструкторе процента заполненности, должна происходить перестройка таблицы: выделение дополнительной памяти под массив элементов, и повторное добавление в новый массив всех существующих элементов

4. Кроме того, необходимо реализовать служебный шаблонный класс Iterator, позволяющий перебрать все входящие в указанную таблицу элементы

5. В классе должно быть реализовано корректное управление динамической памятью: не должно быть утечек памяти, некорректных указателей и т. д

В качестве демонстрационного примера написать программу, которая считывает из файла набор команд для работы с хеш-таблицей, выполняет их и выводит результат.

Входные данные:

В первой строке входного файла заданы два символа,указывающих тип ключа и значения соответственно:
символ **'I'** обозначает тип `int`, **'D'** – `double`, **'S'** – `std::string`

Во второй строке задаётся число **N** – количество команд.

В следующих **N** строках описываются команды.
Команды могут быть двух типов:

- "**A** *key value*" – добавление в хеш-таблицу пары <*key*, *value*>. Если элемент с таким ключом уже есть в хеш-таблице, его значение должно быть обновлено на value

- "**R** *key*" – удаление из хеш-таблицы элемента с ключом *key*, если таковой имеется

Выходные данные:

В выходной файл записать два числа: общее количество элементов в таблице после выполнения команд, и количество "уникальных" элементов в таблице после выполнения команд.

Под "уникальными" понимаются элементы с разными value.

Пример входных и выходных данных:

input.txt	output.txt
I S 5 A 783 Ivanov A 999 Petrova A 986 Sidorov R 986 A 111 Kuznecov	3 3
S D 4 A Ivanov 1.84 A Petrova 1.66 A Ivanov 1.90 A Sidorov 1.90	3 2

Дополнительные задания:

1. Модифицировать свой класс, реализованный в задаче #1, так, чтобы он мог использоваться в качестве ключа в данной хеш-таблице

2. Реализовать шаблонный класс:

```
template <typename K, typename T>
class MultiHashMap {
    ...
};
```

Построенный по тем же принципам, что и HashMap, такой класс поддерживает хранение элементов с одинаковыми ключами.

При этом:

- поиск элемента по ключу возвращает *любой* элемент коллекции с заданным ключом
- удаление по заданному ключу удаляет *все* элементы коллекции с таким ключом

Кроме того, добавляются операции:

- получить все элементы с заданным ключом
- посчитать количество элементов с заданным ключом