Задание на проектирование

Требования к входным данным

Вводятся пользователем через стандартный ввод в виде последовательности строк.

Каждая строка имеет вид “text1 text2 text3”. Поля text1, text2 и text3 разделены символом табуляции, строка заканчивается символом конца строки.

Поле text1 имеет вид IPv4 адресов n1.n2.n3.n4, где n1, n2, n3 и n4 — целые числа от 0 до 255.

Преобразование входных данных

Поля text2 и text3 игнорируются.

Поле text1 преобразуется в IP адрес.

Список IP адресов загружается в память.

Требования к выходные данным

Список IP адресов выводится в стандартный вывод строками вида n1.n2.n3.n4, строка завершается символом окончания строки. Водной строке выводится один адрес.

Содержание выходных данных

Выводится список всех IP адресов в обратном лексикографическом порядке.

За ним без разделителей следует список IP адресов, первый октет которых равен 1, в обратном лексикографическом порядке.

За ним без разделителей следует список IP адресов, первый октет которых равен 46, а второй равен 70, в обратном лексикографическом порядке.

За ним без разделителей следует список IP адресов, любой октет которых равен 46, в обратном лексикографическом порядке.

Требования к реализации

Использование возможностей стандарта C++14.

Возможное содержание тестов

Группа 1. Проверка выделения нужного поля из введённой строки: взято действительно поле text1, поле text1 соответствует шаблону n1.n2.n3.n4

Функционал теста	Проверяемое требование ТЗ
1.1 Сравнение длин полезной и игнорированной частей строк: их сумма должна быть равна длине исходной строки	*Поля text2 и text3 игнорируются*
1.2 Сравнение содержимого полезной и игнорированной частей строк: их объединение должно давать исходную строку	*Поля text2 и text3 игнорируются*
1.3 Проверка реакции на некорректную входную строку: пустая строка, неправильное число полей, отсутствие разделителей полей и строк, несоответствующие разделители полей и строк	*Каждая строка имеет вид “text1 text2 text3”. Поля text1, text2 и text3 разделены символом табуляции, строка заканчивается символом конца строки*

Группа 2. Проверка преобразования выделенного текстового поля в корректный IP адрес

Функционал теста	Проверяемое требование ТЗ
2.1 Проверка правильности определения значения октетов адреса: полученные значения должны быть равны тестовым.	*Поле text1 имеет вид IPv4 адресов n1.n2.n3.n4, где n1, n2, n3 и n4 — целые числа от 0 до 255.* *Поле text1 преобразуется в IP адрес*
2.2 Проверка реакции на некорректное значение поля text1: неправильное число полей, отсутствие разделителей полей, несоответствующие разделители полей	

Группа 3. Проверка сортировки полного списка

Функционал теста	Проверяемое требование ТЗ
3.1 Проверка количества выведенных IP адресов: оно должно соответствовать количеству введённых адресов	*Выводится список всех IP адресов …*
3.2 Проверка правильности сортировки: порядок следования – действительно обратный лексикографический. Например, проверяем, что разность соответствующих октетов текущего и предыдущего адресов не положительна (при условии равенства предыдущих октетов)	*в обратном лексикографическом порядке*

Группа 4. Проверка фильтрации по значению первого октета и последующей сортировки

Функционал теста	Проверяемое требование ТЗ
4.1 Проверка правильности фильтрации: убеждаемся, что в принятых адресах первый октет равен 1.	*За ним без разделителей следует список IP адресов, первый октет которых равен 1 …*
4.2 Проверка правильности фильтрации: убеждаемся, что в отброшенных адресах первый октет не равен 1, количество принятых и отброшенных адресов в сумме равно длине исходного списка.	

Группа 5. Проверка фильтрации по значениям первого и второго октета и последующей сортировки

Функционал теста	Проверяемое требование ТЗ
5.1 Проверка правильности фильтрации: убеждаемся, что в принятых адресах первый октет равен 46, а второй равен 70.	*За ним без разделителей следует список IP адресов, первый октет которых равен 46, а второй равен 70 …*
5.2 Проверка правильности фильтрации: убеждаемся, что в отброшенных адресах первый октет не равен 46, а второй не равен 70, количество принятых и отброшенных адресов в сумме равно длине исходного списка.	

Группа 6. Проверка фильтрации по значению одного любого октета и последующей сортировки

Функционал теста	Проверяемое требование ТЗ
6.1 Проверка правильности фильтрации: убеждаемся, что в принятых адресах какой-то октет равен 46.	*За ним без разделителей следует список IP адресов, любой октет которых равен 46*
6.3 Проверка правильности фильтрации: убеждаемся, что в отброшенных адресах никакой октет не равен 46, количество принятых и отброшенных адресов в сумме равно длине исходного списка.	

Группа 7. Оценка временной сложности

Разбиваем работу приложения на этапы: обработка входных данных, сортировка, фильтрация (функции split, ‘строка – в IP’, filter, filterAny).

Оцениваем время обработки списка из 100, 1000, 10000, 100000, 1000000 строк.

Набираем статистику (для большей достоверности повторяем тесты многократно) по работе всего приложения и отдельных функций, оцениваем вид зависимости времени выполнения от объёма входных данных.

Определяем наиболее затратные этапы обработки, меняем алгоритм, повторяем итерацию. Выбираем оптимальный (по каким-то параметрам: например, время выполнения, требуемая память, сложность кода).