

Задание 1

Задание состоит из двух частей – создание сборочной фермы и реализация приложения для фильтрации IP-адресов. Части почти что независимые – только для сдачи второй потребуется выполнить первую.

Рекомендуемый порядок выполнения – первым этапом разобраться с работой сборочной фермы на простейшем примере (приложение “Hello, World!”), затем использовать настроенную сборочную ферму для построения приложения фильтрации IP-адресов.

Варианты организации исходников по самостоятельным работам:

- отдельный репозиторий под каждую работу
- отдельная ветка в одном и том же репозитории под каждую работу
- отдельная директория в одной и той же ветке одного и того же репозитория

Часть 1. Создание сборочной фермы

Написать программу, выводящую на консоль две строки:

```
build N  
Hello, World!
```

Где вместо N должен выводиться текущий номер сборки. Запустить на этапе сборки тесты, проверяющие валидность номера версии. Выложить исходные тексты в репозиторий на github.

Залогиниться на <https://travis-ci.org/> и включить для своего репозитория автоматическую сборку.

Залогиниться на <https://bintray.com/> (For an Open Source Account и аккаунт github). Создать новый репозиторий - имя произвольное, тип Debian, Default Licenses любая. После создания отредактировать, включив GPG sign. Будет использоваться для публикации всех дальнейших заданий. Создать в репозитории новый пакет - имя произвольное. Ссылка на контроль версии может быть указана как - (знак минус, просто чтобы было заполнено). Так же будет использоваться для публикации последующих работ.

По ссылке <https://bintray.com/profile/edit> в пункте *API Key* скопировать ключ. В настройках репозитория в <https://travis-ci.org> создать переменную, например, *BINTRAY_API_KEY*, и в качестве значения - *API Key*. Это позволит не указывать явно ключ в исходниках.

Самоконтроль

- отсутствие секретов/ключей/паролей в репозитории github
- минимально возможное количество фалов, необходимое для сборки
- репозиторий bintray подписан
- версия пакета увеличивается от сборки к сборке
- актуальная версия выводится в приветственном сообщении
- пакет `helloworld` содержащий исполняемый файл `helloworld` опубликован на bintray

Проверка

Задание считается выполненным успешно, если после подключения репозитория:

```
echo "deb http://dl.bintray.com/porphyryon/otus trusty main" | sudo  
tee -a /etc/apt/sources.list
```

, установки пакета:

```
apt update && apt install -y helloworld
```

, запуска бинарного файла:

```
helloworld
```

появилось сообщение:

```
build N  
Hello, World!
```

Часть 2. Фильтрация IP-адресов

Программа из стандартного ввода читает данные. Данные хранятся построчно. Каждая строка состоит из трех полей, разделенных одним символом табуляции, и завершается символом конца строки. Формат строки:

```
text1 \t text2 \t text3 \n
```

Поля text2 и text3 игнорируются. Поле text1 имеет следующую структуру (ip4 address):

```
n1 . n2 . n3 . n4
```

где n1..4 – целое число от 0 до 255.

Требуется загрузить список ip-адресов в память и отсортировать их в обратном лексикографическом порядке. Пример лексикографической сортировки (по первому числу, затем по второму и так далее):

```
1 . 1 . 1 . 1  
1 . 2 . 1 . 1  
1 . 10 . 1 . 1
```

Соответственно, обратная:

```
1 . 10 . 1 . 1  
1 . 2 . 1 . 1  
1 . 1 . 1 . 1
```

Далее выводим в стандартный вывод следующее:

1. Полный список адресов после сортировки. Одна строка - один адрес.
2. Сразу следом список адресов, первый байт которых равен 1. Порядок сортировки не меняется. Одна строка - один адрес. Списки ничем не разделяются.
3. Сразу продолжается список адресов, первый байт которых равен 46, а второй 70. Порядок сортировки не меняется. Одна строка - один адрес. Списки ничем не разделяются.
4. Сразу продолжается список адресов, любой байт которых равен 46. Порядок сортировки не меняется. Одна строка - один адрес. Списки ничем не разделяются.

Требования к реализации

В приложенном к заданию исходном файле необходимо заменить, где это возможно, конструкции на аналогичные из стандарта C++14. Реализовать недостающий функционал.

Лишний раз проверьте

1. лексикографическая сортировка понятна как для строки, так и для контейнера
2. выбрана соответствующая задаче структура данных

Самопроверка

Макет исходного кода, а также тестовый файл с данными ip_filter.tsv прилагается к материалам занятия. Проверить себя можно следующим образом:

```
cat ip_filter.tsv | ip_filter | md5sum
```

```
24e7a7b2270daee89c64d3ca5fb3da1a -
```