Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет Программной Инженерии и Компьютерной Техники

Дисциплина: Информатика

Лабораторная работа №4
Синтез помехоустойчивого кода
Вариант 23

Выполнил:
Нагорный Николай Викторович
Группа: P3117
Преподаватель:
Марухленко Даниил Сергеевич

Санкт-Петербург, 2024г

Оглавление

Задание

1. Определить номер варианта как остаток деления на 36 последних двух цифр своего идентификационного номера в ISU: например, 125598 / 36 = 26. В случае, если в оба указнных дня недели нет занятий, то увеличить номер варианта на восемь. В случае, если занятий нет и в новом наборе дней, то продолжать увеличивать на восемь.
2. Изучить форму Бэкуса-Наура.
3. Изучить основные принципы организации формальных грамматик.
4. Изучить особенности языков разметки/форматов JSON, YAML, XML.
5. Понять устройство страницы с расписанием на примере расписания лектора:
   https://itmo.ru/ru/schedule/3/125598/raspisanie_zanyatiy.htm
6. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного. При этом необходимо, чтобы хотя бы в одной из выбранных дней было не менее двух занятий (можно использовать своё персональное). В случае, если в данный день недели нет таких занятий, то увеличить номер варианта ещё на восемь.
7. Обязательное задание (позволяет набрать до 45 процентов от максимального числа баллов БаРС за данную лабораторную): написать программу на языке Python 3.x или любом другом, которая бы осуществляла парсинг и конвертацию исходного файла в новый путём простой замены метасимволов исходного формата на метасимволы результирующего формата.
8. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.
9. Дополнительное задание №1 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
   a) Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.
   b) Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.

c) Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.

10. Дополнительное задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
    a) Переписать исходный код, добавив в него использование регулярных выражений.
    b) Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
11. Дополнительное задание №3 (позволяет набрать +25 процентов от максимального числа баллов БаРС за данную лабораторную).
    a) Переписать исходный код таким образом, чтобы для решения задачи использовались формальные грамматики. То есть ваш код должен уметь осуществлять парсинг и конвертацию любых данных, представленных в исходном формате, в данные, представленные в результирующем формате: как с готовыми библиотеками из дополнительного задания №1.
    b) Проверку осуществить как минимум для расписания с двумя учебными днями по два занятия в каждом.
    c) Сравнить полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
12. Дополнительное задание №4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
    a) Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.
    b) Проанализировать полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
13. Дополнительное задание №5 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
    a) Переписать исходную программу, чтобы она осуществляла парсинг и конвертацию исходного файла в любой другой формат (кроме JSON, YAML, XML, HTML): PROTOBUF, TSV, CSV, WML и т.п.

b) Проанализировать полученные результаты, объяснить особенности использования формата. Объяснение должно быть отражено в отчёте.

14. Проверить, что все пункты задания выполнены и выполнены верно.
15. Написать отчёт о проделанной работе.
16. Подготовиться к устным вопросам на защите.

Задания

1. Основное задание

Исходный файл XML:
https://github.com/MiRAX12/Informatics/blob/main/lab4/schedule.xml
Код программы:
https://github.com/MiRAX12/Informatics/blob/main/lab4/Main_task.py
Результат:
https://github.com/MiRAX12/Informatics/blob/main/lab4/schedule.yaml

Программа читает файл построчно записывает в список. Первый цикл берет по очереди эти строки, второй же цикл анализирует эту строку: первый if ищет закрывающий тег и обрезает его. второй if заменяет правую скобку открывающего тега на * (чтобы не изменять длину строки), третий if заменяет левую скобку открывающего тега на &, четвертый if заменяет два пробела на ?^. После завершения второго цикла все вспомогательные символы заменяются. Списки реализуются отдельным условием. Последняя команда первого цикла записать yaml строку.

2. Доп. задание 1

Используется бибиотека xmlplain 1.6.0 для парсинга XML и конвертации в YAML. strip_space – этот флаг удаляет лишние пробелы, не относящиеся к содержанию

Результат отличается от исходного задания

Исходный файл XML:
https://github.com/MiRAX12/Informatics/blob/main/lab4/schedule.xml
Код программы:
https://github.com/MiRAX12/Informatics/blob/main/lab4/Extra1.py
Результат:
https://github.com/MiRAX12/Informatics/blob/main/lab4/scheduleX1.yaml

3. Доп. задание 2

Исходный файл XML:
https://github.com/MiRAX12/Informatics/blob/main/lab4/schedule.xml
Код программы:
https://github.com/MiRAX12/Informatics/blob/main/lab4/Extra2.py

Результат:
https://github.com/MiRAX12/Informatics/blob/main/lab4/scheduleX2.yaml

Программа читает файл полностью и заменяет строчки, найденные по регулярным выражениям на строки yaml. Заданы регулярные выражения для открывающего и закрывающего тега, отдельно для тега day (т.к. нужно учесть атрибут тега и то, что он является списком) и отдельно для тега class (он также является списком), последнее выражение – отступ. В конце программа записывает файл с измененными строками

4. Доп. задание 3

Исходный файл XML:
https://github.com/MiRAX12/Informatics/blob/main/lab4/schedule.xml
Код программы:
https://github.com/MiRAX12/Informatics/blob/main/lab4/Extra3.py
Результат:
https://github.com/MiRAX12/Informatics/blob/main/lab4/scheduleX3.yaml

Программа использует библиотеку re для работы с регулярными выражениями. Также заданы 3 глобальные переменные, что позволяет функциям изменять переменные вне тела функции.

Программа состоит из 4 основных функций:

parse_xml – собирает каждую строку в словарь, разделяя их на name, attributes, text, children, где children – вложенные в текущую строку элементы xml. Функция выделяет нужные элементы с помощью регулярных выражений: main_pattern находит строки, содержащие один полный xml элемент, разбивая его на 3 группы(имя тега, атрибуты, содержимое), attr_pattern получает атрибуты и разбивает их на 2 группы(имя и значение). Алгоритм кода обращается к этим группам и распределяет их по словарю.

5. Доп. задание 4

Код программы:
https://github.com/MiRAX12/Informatics/blob/main/lab4/Extra4.py
Самая быстрая программа – регулярные выражения, далее идут программа через замену и программа через формальные грамматики, их разница незначительна. Самая медленная – с использованием библиотек

Вывод

В ходе выполнения лабораторной работы я научился осуществлять парсинг из XML в YAML при помощи регулярных выражений, библиотек, формальных грамматик и костылей.