

Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
ИТМО»

*Факультет программной инженерии и компьютерной техники
Направление подготовки: 09.03.04 – Программная инженерия,
Системное и прикладное программное обеспечение*

Дисциплина: Информатика

**Лабораторная работа №4
Исследование протоколов,
форматов обмена информацией и языков разметки
документов
Вариант №27**

Выполнил:
Карнажицкий Максим Романович
Группа: Р3111

Проверил:
Доцент факультета ПИиКТ
Малышева Татьяна Алексеевна

г. Санкт-Петербург, 2024

Оглавление

Оглавление.....	2
Задание.....	3
Обязательное задание.....	5
Дополнительное задание №1.....	6
Дополнительное задание №2.....	7
Дополнительное задание №3.....	8
Дополнительное задание №4.....	9
Дополнительное задание №5.....	10
Заключение.....	11
Список литературы.....	12

Задание

1. 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. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.
2. 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.
3. 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.
4. 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.

5. Дополнительное задание №4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
- а. Используя свою исходную программу из обязательного задания и программы из дополнительных заданий, сравнить стократное время выполнения парсинга + конвертации в цикле.
 - б. Проанализировать полученные результаты и объяснить их сходство/различие. Объяснение должно быть отражено в отчёте.
6. Дополнительное задание №5 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
- а. 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.
 - б. Проанализировать полученные результаты, объяснить особенности использования формата. Объяснение должно быть отражено в отчёте.

Обязательное задание

Вариант: YAML -> JSON. Среда, пятница. (27)

Исходный код доступен для просмотра в репозитории на GitHub:
https://github.com/xtern0o/inf_labs/blob/main/lab4/task1/main.py

Результат выполнения программы на примере одного дня можно найти на GitHub-репозитории:

[https://github.com/xtern0o/inf_labs/blob/main/lab4/task1/output_schedule_1day.js
on](https://github.com/xtern0o/inf_labs/blob/main/lab4/task1/output_schedule_1day.json)

Дополнительное задание №1

В качестве готовых библиотек для перевода из `yaml` в `json` я выбрал одноименные модули из стандартной библиотеки Python.

Благодаря их использованию, размер кода сильно сократился.

Исходный код доступен для просмотра на GitHub-репозитории:
https://github.com/xtern0o/inf_labs/blob/main/lab4/task2/main.py

Для чтения YAML-файла я использовал функцию `safe_load()` из библиотеки `yaml`, которая позволяет сохранить результат в словарь – стандартный тип данных для Python.

Для дампа в `json` я использовал функцию `dump()` из библиотеки `json`, которая позволяет конвертировать данные из словаря в формат `json`, записав их сразу же в файл. Помимо этого, я использовал 2 дополнительных параметра:

- `indent=4` – означает, что для отступов (табуляции) в искомом файле следует использовать 4 пробела
- `ensure_ascii=False` – означает, что не нужно представлять символы, которых нет в таблице ASCII в виде экранированных спец. символов вида `\uX`, где `X` – какое-то число. Использовал для сохранения читаемости файла.

Выходные данные, представленные ниже, не отличаются от полученных в первом задании. Разве что, в качестве `indent chars` у стандартной библиотеки используются пробелы, а у меня – символы `\t` табуляции.

Результат выполнения программы можно найти на GitHub-репозитории:
https://github.com/xtern0o/inf_labs/blob/main/lab4/task2/output_schedule_1day.json

Дополнительное задание №2

Файлы результатов программ этого и предыдущих заданий совпадают.

Регулярные выражения были использованы для более простого определения признаков однострочной последовательности, определения однострочных объектов, а так же упростили некоторые условия. Разница в объеме кода несущественна.

Результат выполнения программы можно найти на GitHub-репозитории:
https://github.com/xtern0o/inf_labs/blob/main/lab4/task3/output_schedule_1day.js
[on](#)

Дополнительное задание №3

Код программы, написанный для обязательного задания уже использовал формальные грамматики. То есть, во-первых, при десериализации исходного файла он получал промежуточный результат в виде удобного для анализа типа данных, который является частью языковой модели Python – словаре. А во-вторых, программа может преобразовывать любые корректные файлы формата YAML в JSON. То есть, в ней нет валидатора (реализовывать его – отдельная сложная задача).

Основная функция работает рекурсивно, поочередно “интерпретируя” каждую строку. И в случае, если строка является описанием объекта (или списка), повторно выполняет нужный алгоритм для нее. Выполнив очередную итерацию цикла внутри функции, мы отмечаем строку как посещенную, чтобы в дальнейшем ее не обрабатывать.

Помимо этого, программа умеет обрабатывать однострочные последовательности вида [val1, val2, ...] и однострочные объекты вида {k1: v1, k2: v2, ...}. Для обработки значений используется отдельная функция, которая, в случае последовательности или однострочного объекта, выполняется рекурсивно.

Стоит отметить, что YAML – весьма тяжелый для парсинга язык разметки, и много чего реализовывать, на мой взгляд, не имеет смысла в рамках одной лабораторной работы (в силу сложности). Например, anchors (якоря и ссылки), поддержка мультдокументов, явное приведение типов данных.

Исходный код программы:
https://github.com/xtern0o/inf_labs/blob/main/lab4/task4/main.py

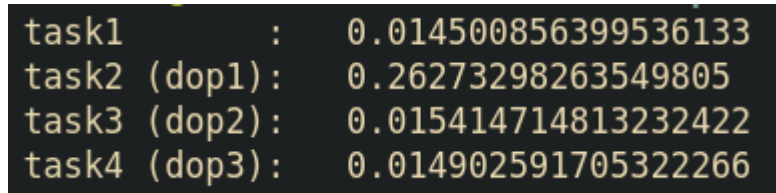
Добавление второго дня в расписание не повлияло на работоспособность программы: https://github.com/xtern0o/inf_labs/blob/main/lab4/task4/output.json.

Выходные данные полностью совпадают с файлом из доп. задания №1, что говорит о корректности перевода.

Дополнительное задание №4

Исходный код программы:
https://github.com/xtern0o/inf_labs/blob/main/lab4/task5.py

Вывод программы представлен на скриншоте ниже.



```
task1      : 0.014500856399536133
task2 (dop1): 0.26273298263549805
task3 (dop2): 0.015414714813232422
task4 (dop3): 0.014902591705322266
```

Рисунок 5.1 – вывод программы, измеряющей стократное выполнение программ из предыдущих заданий

Таким образом, самое медленное время выполнения показал вариант программы, использующий готовые библиотеки. Самым же быстрым оказался вариант, использующий формальные грамматики.

Вариант, использующий регулярные выражения, в среднем, оказывается незначительно медленнее. Вероятно, это связано с тем, что каждое новое регулярное выражение приходится заново компилировать.

Дополнительное задание №5

В качестве формата, в который я буду конвертировать данные я выбрал CSV (comma separated values). Но вместо запятой я буду использовать в качестве сепаратора “;”, так как в адресе входной строки содержатся запятые (это тоже считается форматом CSV)

Для этого задания я использовал написанный ранее алгоритм десериализации YAML-файла в словарь, после чего переводил в формат csv.

Стоит отметить, что файл работает корректно только для файла со структурой расписания. Задание не требует использования формальных грамматик для работы с любыми файлами.

Исходный

код:

https://github.com/xtern0o/inf_labs/blob/main/lab4/task6/main.py

Полученный

результат:

https://github.com/xtern0o/inf_labs/blob/main/lab4/task6/output.csv

Заключение

В ходе выполнения лабораторной работы я очень близко познакомился с такими форматами разметки, как YAML и JSON. Написал парсер из первого типа в другой, используя формальные грамматики. Закрепил навыки в области написания регулярных выражений.

Список литературы

1. Балакшин П. В., Соснин В. В. Информатика: методическое пособие. – г. Санкт-Петербург: Университет ИТМО, 2015 – Режим доступа: <https://picloud.pw/media/resources/posts/2018/02/19/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B0.pdf>
2. Балакшин П.В., Соснин В.В., Калинин И.В., Малышева Т.А., Раков С.В., Рущенко Н.Г., Дергачев А.М. Информатика: лабораторные работы и тесты. – СПб: Университет ИТМО, 2019. – 56 с. – Режим доступа: <https://books.ifmo.ru/file/pdf/2464.pdf>