Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

**Отчет по лабораторной работе №4**

**"Исследование протоколов,**

**форматов обмена информацией и языков разметки**

**документов"**

**По дисциплине**

**«Информатика»**

**Вариант 20**

Выполнил: студент группы P3115

Лазеев Сергей Максимович

Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург

2021

Оглавление

[Задание 3](#_Toc85068247)

[Решение 4](#_Toc85068248)

[Вывод 6](#_Toc85068249)

[Список используемой литературы 7](#_Toc85068250)

Задание

1. Обязательное задание (позволяет набрать до 65 процентов от максимального числа баллов БаРС за данную лабораторную): написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в новый. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.
2. Дополнительное задание 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.

1. Дополнительное задание 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.

1. Дополнительное задание задание №3 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).

a) Используя свою исходную программу из обязательного задания, программу из дополнительного задания №1 и программу из дополнительного задания №2, сравнить десятикратное время выполнения парсинга + конвертации в цикле.

b) Проанализировать полученные результаты и объяснить их сходство/различие.

1. Дополнительное задание задание №4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную.

a) Переписать исходную, 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 осоебнности использованного формата.

Решение

Алгоритмы для всех заданий были написаны мною на языке программирования Python. Все 4 файла с решениями были выложены мною на сайт github (ссылка - <https://github.com/k1b24/ICT_lab4>). Python-файл с решением представляет из себя алгоритм решения задания, который на основе входного файла создает файл с аналогичным названием, но с другим расширением куда помещает преобразованный входной файл. В первом задании мною было выполнено преобразование из формата xml в формат json без использования сторонних библиотек и регулярных выражений. Во втором задании мною было выполнено преобразование с использованием библиотек xmltodict и json. В третьем задании мною был изменен код первого задания при помощи использования регулярных выражений. В четвертом задании с помощью библиотеки time для python мною было измерено десятикратное время выполнения всех трех, написанных мною парсеров. В пятом задании мною был описан алгоритм преобразования исходного xml файла в csv таблицу.

Код для задания №1:

*def my\_parse():*

*# Функция для нахождения повторяющихся тегов, чтобы эти теги выносить в []*

*def find\_repitable\_tags(m):*

*repitable\_tags = {}*

*for i in range(len(m)):*

*tag\_name = m[i][m[i].find('<')+1:m[i].find('>')].strip()*

*if m.count(m[i]) != 1 and '</' not in m[i]:*

*if tag\_name not in repitable\_tags:*

*repitable\_tags[tag\_name] = 1*

*else:*

*repitable\_tags[tag\_name] += 1*

*return repitable\_tags*

*#открываем xml файл*

*xml\_file = open('Schedule.xml', 'r', encoding='UTF-8')*

*xml\_lines = xml\_file.readlines()*

*xml\_lines = xml\_lines[1::]*

*xml\_string = ''*

*for i in range(0, len(xml\_lines)):*

*xml\_string += xml\_lines[i].lstrip().rstrip()*

*#создаем json файл*

*json\_file = open('Schedule.json', 'w', encoding='UTF-8')*

*tabs = 0 #переменная отвечающая за отступы*

*json\_lines = ['{'] #массив в котором каждый элемент отвечает за каждую строку в финальном файле*

*tabs += 1*

*repitable\_tags = find\_repitable\_tags(xml\_lines) # словарь отвечающий за повторяющиеся теги для установки []*

*used\_repitable\_tags = [] #список хранящий в себе уже использованные в качестве открывающей [ теги*

*prev\_is\_content = False #переменная отвечающая за проверку предыдущей проходки по циклу, если в предыдущей проходке мы вычленяли контент, то она True, иначе False*

*while len(xml\_string) != 0: #идем по строке с xml кодом и постепенно ее уменьшаем*

*if xml\_string[:2] == '</': #обработка строки если в ней встретился закрывающий тег*

*nexttag = xml\_string[3:]*

*nexttag = nexttag[nexttag.find('<')+1:]*

*nexttag = nexttag[:nexttag.find('>')] # находим следующий тег*

*tabs -= 1*

*if not(prev\_is\_content): # если предыдущая проходка не обрабатывала контент*

*tag = xml\_string[2:xml\_string.find('>')] #текущий обрабатываемый тег*

*if tag in repitable\_tags and repitable\_tags[tag] == 0 and nexttag != tag: #проверка тега на его принадлежность к повторяющемуся тегу*

*s = ' ' \* tabs + '}\n'*

*tabs -= 1*

*s += ' ' \* tabs + ']'*

*json\_lines.append(s)*

*else:*

*json\_lines.append(' ' \* tabs + '}')*

*if nexttag[0] != '/' and xml\_string.count('>') > 1: #если следующий тег открывающийся*

*json\_lines[len(json\_lines)-1] += ','*

*prev\_is\_content = False*

*r = xml\_string.find('>')*

*xml\_string = xml\_string[r+1:]*

*elif xml\_string[0] == '<': #обработка строки если в ней встретился открывающий тег*

*r = xml\_string.find('>')*

*tag = xml\_string[1:r]*

*if tag not in repitable\_tags: #если тег не повторяющийся*

*s = ' ' \* tabs + '"' + tag + '": '*

*if xml\_string[r+1] == '<':*

*s += '{'*

*elif tag not in used\_repitable\_tags: #иначе если тег повторяющийся и не использовалься*

*s = ' ' \* tabs + '"' + tag + '": [ \n'*

*tabs += 1*

*s += ' ' \* tabs + '{'*

*used\_repitable\_tags.append(tag)*

*repitable\_tags[tag] -= 1*

*elif repitable\_tags[tag] >= 1: #если использовался, но сейчас будет использоваться не в последний раз*

*s = ' ' \* tabs + '{'*

*repitable\_tags[tag] -= 1*

*xml\_string = xml\_string [r+1:]*

*tabs += 1*

*json\_lines.append(s)*

*else: # #обработка строки во всех других случаях*

*r = xml\_string.find('<')*

*content ='"' + xml\_string[:r].strip() + '"'*

*xml\_string = xml\_string[r:]*

*json\_lines[len(json\_lines)-1] += content*

*prev\_is\_content = True*

*json\_lines.append('}')*

*for elem in json\_lines:*

*json\_file.write(elem+'\n')*

*my\_parse()*

Код для задания №2:

*def parse\_with\_lib():*

*import xmltodict*

*import json*

*xml\_file = open('Schedule.xml', 'r', encoding='UTF-8')*

*json\_file = open('Schedule.json', 'w', encoding='UTF-8')*

*xml\_content = xml\_file.read()*

*dict = xmltodict.parse(xml\_content, encoding='UTF-8')*

*print(json.dumps(dict, sort\_keys=False, indent=4, ensure\_ascii=False, separators=(',', ': ')), file=json\_file)*

*parse\_with\_lib()*

Код для задания №3:

*def my\_parse\_with\_re():*

*import re*

*# Функция для нахождения повторяющихся тегов, чтобы эти теги выносить в []*

*def find\_repitable\_tags(m):*

*repitable\_tags = {}*

*for i in range(len(m)):*

*tag\_name = m[i][m[i].find('<')+1:m[i].find('>')].strip()*

*if m.count(m[i]) != 1 and '</' not in m[i]:*

*if tag\_name not in repitable\_tags:*

*repitable\_tags[tag\_name] = 1*

*else:*

*repitable\_tags[tag\_name] += 1*

*return repitable\_tags*

*#открываем xml файл*

*xml\_file = open('Schedule.xml', 'r', encoding='UTF-8')*

*xml\_lines = xml\_file.readlines()*

*xml\_lines = xml\_lines[1::]*

*xml\_string = ''*

*for i in range(0, len(xml\_lines)):*

*xml\_string += xml\_lines[i].lstrip().rstrip()*

*#создаем json файл*

*json\_file = open('Schedule.json', 'w', encoding='UTF-8')*

*tabs = 0 #переменная отвечающая за отступы*

*json\_lines = ['{'] #массив в котором каждый элемент отвечает за каждую строку в финальном файле*

*tabs += 1*

*repitable\_tags = find\_repitable\_tags(xml\_lines) # словарь отвечающий за повторяющиеся теги для установки []*

*used\_repitable\_tags = [] #список хранящий в себе уже использованные в качестве открывающей [ теги*

*prev\_is\_content = False #переменная отвечающая за проверку предыдущей проходки по циклу, если в предыдущей проходке мы вычленяли контент, то она True, иначе False*

*close\_pattern = re.compile(r"<\/.+>$")# паттерн отвечающий за закрывающий тег*

*open\_pattern = re.compile(r"<[^\/]+>$")# паттерн отвечающий за открывающий тег*

*while len(xml\_string) != 0: #идем по строке с xml кодом и постепенно ее уменьшаем*

*if bool(re.match(close\_pattern, xml\_string[:xml\_string.find('>')+1])): #обработка строки если в ней встретился закрывающий тег*

*nexttag = xml\_string[3:]*

*nexttag = nexttag[nexttag.find('<'):]*

*nexttag = nexttag[:nexttag.find('>')+1] # находим следующий тег*

*tabs -= 1*

*if not(prev\_is\_content): # если предыдущая проходка не обрабатывала контент*

*tag = xml\_string[2:xml\_string.find('>')] #текущий обрабатываемый тег*

*if tag in repitable\_tags and repitable\_tags[tag] == 0 and nexttag != tag: #проверка тега на его принадлежность к повторяющемуся тегу*

*s = ' ' \* tabs + '}\n'*

*tabs -= 1*

*s += ' ' \* tabs + ']'*

*json\_lines.append(s)*

*else:*

*json\_lines.append(' ' \* tabs + '}')*

*if bool(re.match(open\_pattern, nexttag)) and xml\_string.count('>') > 1: #если следующий тег открывающийся*

*json\_lines[len(json\_lines)-1] += ','*

*prev\_is\_content = False*

*r = xml\_string.find('>')*

*xml\_string = xml\_string[r+1:]*

*elif bool(re.match(open\_pattern, xml\_string[:xml\_string.find('>')+1])): #обработка строки если в ней встретился открывающий тег*

*r = xml\_string.find('>')*

*tag = xml\_string[1:r]*

*if tag not in repitable\_tags: #если тег не повторяющийся*

*s = ' ' \* tabs + '"' + tag + '": '*

*nexttag = xml\_string[r+1:]*

*nexttag = nexttag[:nexttag.find('>')+1]*

*if bool(re.match(open\_pattern, nexttag)): #является ли следующая строка тегом или нет*

*s += '{'*

*elif tag not in used\_repitable\_tags: #иначе если тег повторяющийся и не использовалься*

*s = ' ' \* tabs + '"' + tag + '": [ \n'*

*tabs += 1*

*s += ' ' \* tabs + '{'*

*used\_repitable\_tags.append(tag)*

*repitable\_tags[tag] -= 1*

*elif repitable\_tags[tag] >= 1: #если использовался, но сейчас будет использоваться не в последний раз*

*s = ' ' \* tabs + '{'*

*repitable\_tags[tag] -= 1*

*xml\_string = xml\_string [r+1:]*

*tabs += 1*

*json\_lines.append(s)*

*else: # #обработка строки во всех других случаях*

*r = xml\_string.find('<')*

*content ='"' + xml\_string[:r].strip() + '"'*

*xml\_string = xml\_string[r:]*

*json\_lines[len(json\_lines)-1] += content*

*prev\_is\_content = True*

*json\_lines.append('}')*

*for elem in json\_lines:*

*json\_file.write(elem+'\n')*

*my\_parse\_with\_re()*

Код для задания №4

*from Task2 import my\_parse\_with\_re*

*from Task1 import parse\_with\_lib*

*from Task0 import my\_parse*

*import time*

*my\_time = time.time()*

*for i in range(10):*

*my\_parse()*

*my\_time = time.time() - my\_time*

*my\_re\_time = time.time()*

*for i in range(10):*

*my\_parse\_with\_re()*

*my\_re\_time = time.time() - my\_re\_time*

*libs\_time = time.time()*

*for i in range(10):*

*parse\_with\_lib()*

*libs\_time = time.time() - libs\_time*

*print('Парсер без готовых библиотек: ' + str(my\_time))*

*print('Парсер без готовых библиотек с регулярными выражениями: ' + str(my\_re\_time))*

*print('Парсер с готовыми библиотеками: ' + str(libs\_time))*

*if libs\_time > min(my\_time, my\_re\_time):*

*if my\_time < my\_re\_time:*

*print("Парсер без готовых библиотек быстрее.")*

*else:*

*print("Парсер без готовых библиотек с регулярными выражениями быстрее.#")*

*else:*

*print("Парсер с готовыми библиотеками быстрее.")*

Код для задания №5:

*from xml.etree import ElementTree*

*import csv*

*# парсим xml*

*xml = ElementTree.parse("Schedule.xml")*

*# создаем csv файл*

*csvfile = open("Schedule.csv",'w', newline='', encoding='utf-8')*

*csvfile\_writer = csv.writer(csvfile)*

*# добавляем заголовки в csv файл*

*csvfile\_writer.writerow(["week","time","place", "room", "name", "teacher", "lesson-format"])*

*for schedule in xml.findall("schedule"):*

*for day in schedule.findall("day"):*

*for lessons in day.findall("lessons"):*

*for lesson in lessons.findall("lesson"):*

*if(lesson):*

*#извлечь данные о парах*

*week = lesson.find("week")*

*time = lesson.find("time")*

*place = lesson.find("place")*

*room = lesson.find("room")*

*name = lesson.find("name")*

*teacher = lesson.find("teacher")*

*lesson\_format = lesson.find("lesson-format")*

*csv\_line = [week.text.strip().replace(',', ';'), time.text.strip(), place.text.strip().replace(',', ';'), room.text.strip(), name.text.strip(), teacher.text.strip(), lesson\_format.text.strip()]*

*#добавить строку в файл*

*csvfile\_writer.writerow(csv\_line)*

Исходный xml файл:

<?xml version="1.0" encoding="utf-8"?>

<root>

<title> Расписание группы P3115 </title>

<schedule>

<day>

<short-name-of-day> Чт </short-name-of-day>

<lessons>

<lesson>

<week> 3, 5, 7, 9, 11, 13, 15, 17 </week>

<time> 10:00-11:30 </time>

<place> Кронверкский пр., д.49, лит.А </place>

<room> 369А ауд. </room>

<name> Дискретная математика (прак) </name>

<teacher> Поляков Владимир Иванович </teacher>

<lesson-format> Очно - дистанционный </lesson-format>

</lesson>

<lesson>

<week> 3, 5, 7, 9, 11, 13, 15, 17 </week>

<time> 11:40-13:10 </time>

<place> Кронверкский пр., д.49, лит.А </place>

<room> 369А ауд. </room>

<name> Дискретная математика (прак) </name>

<teacher> Поляков Владимир Иванович </teacher>

<lesson-format> Очно - дистанционный </lesson-format>

</lesson>

<lesson>

<week> 3, 5, 7, 9, 11, 13, 15, 17 </week>

<time> 13:30-15:00 </time>

<place> Кронверкский пр., д.49, лит.А </place>

<room> 285 ауд. </room>

<name> Дискретная математика (лек) </name>

<teacher> Поляков Владимир Иванович </teacher>

<lesson-format> Очно - дистанционный </lesson-format>

</lesson>

</lessons>

</day>

</schedule>

</root>

Полученный json файл (все программы создают идентичный json файл):

{

"root": {

"title": "Расписание группы P3115",

"schedule": {

"day": {

"short-name-of-day": "Чт",

"lessons": {

"lesson": [

{

"week": "3, 5, 7, 9, 11, 13, 15, 17",

"time": "10:00-11:30",

"place": "Кронверкский пр., д.49, лит.А",

"room": "369А ауд.",

"name": "Дискретная математика (прак)",

"teacher": "Поляков Владимир Иванович",

"lesson-format": "Очно - дистанционный"

},

{

"week": "3, 5, 7, 9, 11, 13, 15, 17",

"time": "11:40-13:10",

"place": "Кронверкский пр., д.49, лит.А",

"room": "369А ауд.",

"name": "Дискретная математика (прак)",

"teacher": "Поляков Владимир Иванович",

"lesson-format": "Очно - дистанционный"

},

{

"week": "3, 5, 7, 9, 11, 13, 15, 17",

"time": "13:30-15:00",

"place": "Кронверкский пр., д.49, лит.А",

"room": "285 ауд.",

"name": "Дискретная математика (лек)",

"teacher": "Поляков Владимир Иванович",

"lesson-format": "Очно - дистанционный"

}

]

}

}

}

}

}

Результат работы программы, сравнивающей время работы всех трёх написанных мною парсеров, представлен на рисунке №1:

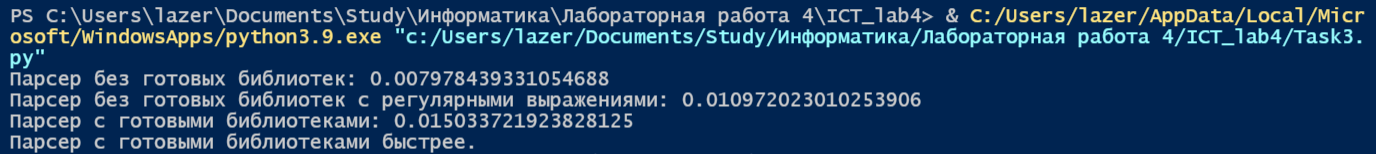
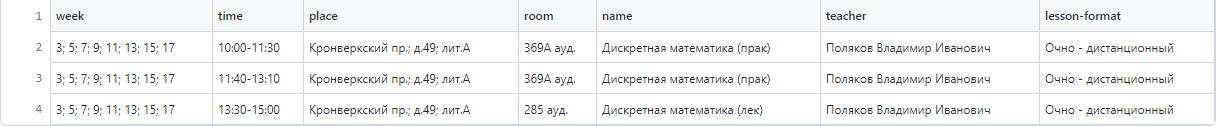


Рисунок №1 – результат работы программы, сравнивающей парсеры из первого, второго и третьего задания.

Результат работы программы, выполняющей преобразование xml файла в csv таблицу представлен на рисунке №2:

  
Рисунок №2 – таблица, полученная в результате преобразования исходного xml файла в csv формат.

Вывод

В процессе выполнения данной лабораторной работы я попробовал вручную написать алгоритм осуществляющий парсинг и преобразование xml файла в json и csv файл. Описанный мной алгоритм преобразования из xml в json с использованием регулярных выражений оказался медленнее, чем решение с использованием готовых библиотек. Вероятно, это связано с несовершенством описанного мной алгоритма.

Список используемой литературы

1. Орлов С. А., Цилькер Б. Я. Организация ЭВМ и систем: Учебник для вузов. 2-е изд. – СПб.: Питер, 2011. – 688 с.: ил. - Режим доступа: URL: <https://bit.ly/3kxdN1m>
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.  
   (<https://books.ifmo.ru/file/pdf/2464.pdf>)